Evolutionary Optimization of Low-Discrepancy Sequences

François-Michel De Rainville, Christian Gagné, Olivier Teytaud, Denis Laurendeau

EVOLUTIONARY OPTIMIZATION OF LOW-DISCREPANCY SEQUENCES

FRANÇOIS-MICHEL DE RAINVILLE, CHRISTIAN GAGNÉ, OLIVIER TEYTAUD, AND DENIS LAURENDEAU

Abstract. Low-discrepancy sequences provide a way to generate quasi-random numbers of high dimensionality with a very high level of uniformity. The nearly orthogonal Latin hypercube and the generalized Halton sequence are two popular methods when it comes to generate low-discrepancy sequences. In this article, we propose to use evolutionary algorithms in order to find optimized solutions to the combinatorial problem of configuring generators of these sequences. Experimental results show that the optimized sequence generators behave at least as well as generators from the literature for the Halton sequence and significantly better for the nearly orthogonal Latin hypercube.

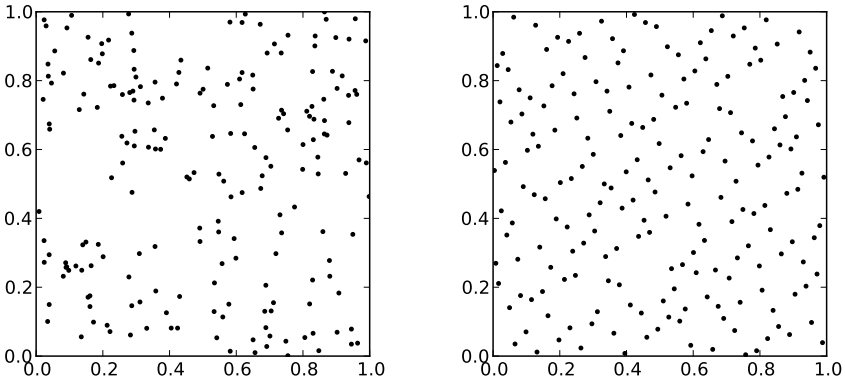
1. Introduction

Many fields of research such as optimization, learning, design of experiments, and numerical integration use randomly generated samples to explore complex multi-dimensional spaces. The uniformity of the generated samples is usually important as it is, very often, strongly correlated with the accuracy of the insight gained by the exploration (or the accuracy of the estimation) [Niederreiter 1978]. Pseudo-random number generators are often unsatisfying with respect to the uniformity of the produced distributions (Figure 1(a)). It is very uncommon for them to generate samples distributed as evenly as the ones produced by a quasi-random number generator (Figure 1(b)). In many cases, the standard random number generator can be replaced easily in order to explore more efficiently the problem's complex space. As Niederreiter [1978] states:

> [...] instead of trying to cope with the impalpable concept of randomness, one should select points according to a deterministic scheme that is well suited for the problem at hand. [...] For instance, in the area of numerical integration it turns out to be quite irrelevant whether the sample points or "nodes" are truly random; of primary importance is really the even distribution of the points over [the integration domain].

These quasi-random methods have already been used in many areas of computer science; see, for example, Niederreiter [1992], L'Ecuyer and Lemieux [2000], Cervellera and Muselli [2004], Glasserman [2004], Auger et al. [2006], L'Ecuyer [2009], and Lemieux [2009].

Evolutionary algorithms [Holland 1975] have long been recognized as powerful meta-heuristic optimization techniques and have proved multiple times their ability to optimize very difficult problems. However, these algorithms (and stochastic optimization methods in general) have not been used to configure low-discrepancy sequence generators.



(a) Pseudo-Random (b) Quasi-Random

FIGURE 1. The uniformity of some pseudo-randomly generated samples (a) can be significantly poorer than quasi-randomly generated ones (b).

The structure of the article is as follows. First, some discrepancy measures are presented in Section 2, which are used to assess and compare the samples distribution quality obtained with quasi-random number generators. Then, some background is given on the nearly orthogonal Latin hypercube [Cioppa and Lucas 2007] (Section 3), providing good sampling of some space with a minimum number of points (i.e., design of experiments), and the generalized Halton sequence [Braaten and Weller 1979] (Section 4), useful to generate low-discrepancy sequences with an unbounded number of points. The proposed method for optimizing the low-discrepancy number generator based on evolutionary algorithms is then presented in Section 5. Experimental results are reported in Section 6, comparing results obtained with the evolutionary-based optimization to state-of-the-art methods. Finally, Section 7 presents a comparison over several test functions between the generalized Halton sequence's configuration found by the evolutionary algorithm and some well-known configurations.

2. DISCREPANCY

In many problems, it is nearly impossible to evaluate exactly the value of a multidimensional integral over the s-dimensional unit cube $C^s = [0,1)^s$ of the form

$$I(f) = \int_{C^s} f(x)dx. \tag{1}$$

In these cases, the problem can only be approximated by the sample mean

$$\hat{I}_N(f) = \frac{1}{N}\sum_{z\in\mathcal{S}} f(z), \tag{2}$$

where $\mathcal{S}$ is a finite point set with each sample in $[0,1)^s$. The more the samples from $\mathcal{S}$ are uniformly distributed over $[0,1)^s$ the greater are the chances of having

an accurate estimation of the integral. The absolute error bound $E_N(f)$ on the estimation takes the form

$$E_N(f) = |I(f) - \hat{I}_N(f)| \leq D(\mathcal{S})V(f),$$

where $D(\mathcal{S})$ is a measure of nonuniformity of $\mathcal{S}$ and $V(f)$ is a measure of fluctuation of f.

The discrepancy measures by how much the empirical distribution of the finite point set $\mathcal{S}$ deviates from the uniform distribution in $[0,1)^s$. The star discrepancy is defined by Niederreiter [1992] as

$$D^*(\mathcal{S}) = \sup_{x \in C^s} \left| \frac{\text{card}\,(\mathcal{S} \cap [0,x))}{N} - \text{Vol}([0,x)) \right|, \tag{3}$$

where $\text{card}(\mathcal{A})$ is the cardinal of the point set $\mathcal{A}$ and $\text{Vol}([0,x))$ is the volume of the rectangular box with opposite corners at 0 and x. Then, the Kosma-Hlakwa inequality gives a precise definition of the absolute error bound for the star discrepancy. It is defined as

$$E_N(f) = |I(f) - \hat{I}_N(f)| \leq D^*(\mathcal{S})V(f), \tag{4}$$

where $V(f)$ is the Hardy-Krause variation in s dimensions of the function f. When $V(f) < \infty$, (4) states that an error bound on the estimation of f may be computed from a discrepancy measure on the point set $\mathcal{S}$. A definition of low-discrepancy sequences can be drawn from that inequality, a low-discrepancy sequence shall provide an estimation $\hat{I}_N(f)$ of $I(f)$ so that the error $E_N(f)$ is bounded by $O(N^{-1}(\log N)^s)$.

The calculation of $D^*(\mathcal{S})$ implies computing the volume of every rectangular box with one corner anchored at the origin and the other at either a point $x \in \mathcal{S}$ or $y \notin \mathcal{S}$ implied by the supremum operator. In real life, this is computationally very expensive for point sets of dimensionality $s > 2$ and more than a few points (small N). The supremum norm in (3) may be replaced by an L_2-norm in order to reduce the complexity. Warnock [1972] gives the following equation for the L_2-star discrepancy of a set containing N points, with $\mathcal{S}_{x,y}$ being the y^{th} coordinate of the x^{th} point of the point set $\mathcal{S}$.

$$\begin{aligned} T^*_{2,N}(\mathcal{S})^2 = 3^{-s} &+ \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \prod_{k=1}^{s} [1 - \max(\mathcal{S}_{i,k}, \mathcal{S}_{j,k})] \\ &- \frac{2^{1-s}}{N} \sum_{i=1}^{N} \prod_{k=1}^{s} (1 - \mathcal{S}_{i,k}^2) \end{aligned} \tag{5}$$

Hickernell [1998] gives different versions of the discrepancy using an L_2-norm and different anchor points, one of them is the modified L_2 discrepancy

$$\begin{aligned} M_{2,N}(\mathcal{S})^2 = \left(\frac{4}{3}\right)^s &+ \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \prod_{k=1}^{s} [2 - \max(\mathcal{S}_{i,k}, \mathcal{S}_{j,k})] \\ &- \frac{2^{1-s}}{N} \sum_{i=1}^{N} \prod_{k=1}^{s} (3 - \mathcal{S}_{i,k}^2). \end{aligned} \tag{6}$$

The modified L_2 discrepancy is used here instead of the original discrepancy since the original one has the undesirable property of decreasing as the dimension increases for a uniform random sampling [Morokoff and Caflisch 1994; Hickernell 1998].

TABLE 1. Latin Hypercube Samplings in Two Dimensions

(a) LH

A	B
1	2
2	4
3	5
4	3
5	1

(b) LH

A	B
1	5
2	4
3	3
4	1
5	2

(c) OLH

A	B
1	2
2	5
3	3
4	1
5	4

(a) and (b) are random LH sampling with (b) being degenerated with strong correlation, (c) is an OLH sampling constructed using Ye's technique. Correlation between factor A and B is respectively $-0.3, -0.9$, and 0.0 for the three samplings.

3. Nearly Orthogonal Latin Hypercube

A Nearly Orthogonal Latin Hypercube (NOLH) is a sampling method, often used in design of experiments, that shows good bidimensional space-filling properties for a relatively small number of samples. It is also characterized by a low correlation between the factors of each sample, which is beneficial when regression analysis is applied. The NOLH is based on the Latin Hypercube (LH) sampling [McKay et al. 1979], which consists, in the two-dimensional case, in a grid containing one and only one sample in each of its rows and columns. In higher dimensions, LH samplings are built based on the number of levels by factor needed, each point being a permutation of these levels. Table 1 presents two different possible LH samplings of four samples for two variables, A and B. One problem with the LH technique is that it may lead to some ill-conditioned samplings, as in Table 11(b) (see the correlation). In order to circumvent this disadvantage, Ye [1998] introduced a construction algorithm that ensures no correlation between each pair of variables of its orthogonal LH (OLH) (Table 11(c)). The NOLH construction algorithm is a modification, by Cioppa and Lucas [2007], of Ye's algorithm that allows spaces of higher dimensionality to be explored with less samples. Cioppa and Lucas [2007] also showed that by allowing small correlation between the factors, design uniformity could increase dramatically. Cioppa and Lucas' algorithm is the groundwork of the current part of the article and is thus described in detail in the remainder of this section.

3.1. NOLH Construction.

The NOLH construction begins with the base vector $\mathbf{e}$, from which matrices $\mathbf{M}$ and $\mathbf{S}$ and the set of matrices $\mathcal{A}$ are built. The base vector $\mathbf{e}$ is a permutation of $[1\ 2\ \cdots\ q]$, where q is the number of positive levels of the NOLH, $\mathbf{e}$ being the only parameter that needs to be set in order to build an NOLH. The number of samples n and the dimensionality s of an NOLH are obtained from the order parameter m according to

$$n = 2^m + 1, \tag{7}$$

$$s = m + \binom{m-1}{2}, \tag{8}$$

$$q = \frac{n-1}{2} = 2^{m-1}. \tag{9}$$

TABLE 2. NOLH Parameters by Design Order

Order (m)	Dimensionality (s)	Samples (n)
4	7	17
5	11	33
6	16	65
7	22	129

Table 2 presents these variables for $m \in \{4, 5, 6, 7\}$.

First, the collection $\mathcal{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_{m-1}\}$ contains $m-1$ permutation matrices of size $q \times q$ that are obtained from

$$\mathbf{A}_i = \underbrace{\mathbf{I} \otimes \cdots \otimes \mathbf{I}}_{m-1-i} \otimes \underbrace{\mathbf{R} \otimes \cdots \otimes \mathbf{R}}_{i}, \qquad i = 1, \ldots, m-1,$$

where $\mathbf{I}$ is the 2×2 identity matrix, $\mathbf{R}$ is the 2×2 $\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$ matrix, and $\otimes$ is the Kronecker product of matrices.

Then, the columns $\mathbf{m}_1$ to $\mathbf{m}_s$ of matrix $\mathbf{M}$ are given by

$$\mathbf{m}_i = \begin{cases} \mathbf{e} & \text{for} \quad i = 1 \\ \mathbf{A}_j \mathbf{e} & \text{for} \quad i = 2, \ldots, m \\ \mathbf{A}_k \mathbf{A}_l \mathbf{e} & \text{for} \quad i = m+1, \ldots, s \end{cases},$$

where $j = 1, \ldots, m-1$ and k and l are the first and second element of every 2-combination of $\{1, \ldots, m-1\}$.

Matrix $\mathbf{M}$ with $\mathbf{e}$ set to the identity permutation and $m = 4$ is given in (10).

$$\mathbf{M} = \begin{bmatrix} 1 & 2 & 4 & 8 & 3 & 7 & 5 \\ 2 & 1 & 3 & 7 & 4 & 8 & 6 \\ 3 & 4 & 2 & 6 & 1 & 5 & 7 \\ 4 & 3 & 1 & 5 & 2 & 6 & 8 \\ 5 & 6 & 8 & 4 & 7 & 3 & 1 \\ 6 & 5 & 7 & 3 & 8 & 4 & 2 \\ 7 & 8 & 6 & 2 & 5 & 1 & 3 \\ 8 & 7 & 5 & 1 & 6 & 2 & 4 \end{bmatrix} \tag{10}$$

Thereafter, matrix $\mathbf{S}$ contains the $+\mathbf{1}$ vector in its first column. The $m-1$ subsequent columns $\mathbf{s}_2, \ldots, \mathbf{s}_m$ reflect a 2^{m-1} full factorial design.

The remaining columns are given by

$$\mathbf{s}_i = \mathbf{s}_j \mathbf{s}_k,$$

where $i = m+1, \ldots, s$ and j and k are the first and second element of every 2-combination of $\{1, \ldots, m-1\}$.

Matrix $\mathbf{S}$ with $m = 4$ is presented in (11).

$$\mathbf{S} = \begin{bmatrix} +1 & -1 & -1 & -1 & +1 & +1 & +1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 \\ +1 & +1 & +1 & -1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & +1 & -1 & +1 & -1 & +1 & -1 \\ +1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & +1 & +1 & +1 & +1 & +1 & +1 \end{bmatrix} \tag{11}$$

Finally, matrix $\mathbf{T}$ is obtained from the Hadamard product of matrices $\mathbf{M}$ and $\mathbf{S}$. The final NOLH sampling $\mathcal{S}_{\text{NOLH}}$ is derived from matrix $\mathbf{T}$ by adding to it a null central point and its negative mirror image, as in (12).

$$\mathbf{T} = \mathbf{M} \cdot \mathbf{S}$$

$$\mathcal{S}_{\text{NOLH}} = \begin{bmatrix} \mathbf{T} \\ \mathbf{0}^T \\ -\mathbf{T} \end{bmatrix} \tag{12}$$

In order to obtain an NOLH sampling of dimensions between orders, one can remove the columns of the final design obtained from the superior order giving the best properties. For example, an NOLH sampling of dimension 8 is built from an NOLH sampling of order 5 (in 11 dimensions), from which 3 columns have been removed.

3.2. **Florian's Method.** Florian's method is a routine that is applied to a sampling matrix in order to decrease its maximum pairwise correlation. Briefly, it exchanges the position of two levels within the same column of the sampling matrix until no more improvement of the correlation in the matrix can be achieved. Cioppa and Lucas [2007] use this procedure only on the sampling matrices that pass a certain screening test. In the current article, when Florian's method is used, it is applied to every sampling matrix without discrimination. Interested readers are referred to Cioppa and Lucas [2007] and Florian [1992] for more details on Florian's method.

4. Quasi-Random Sequence

Quasi-random number generators produce low-discrepancy multidimensional point sets. Such generators may be very interesting to functionally replace pseudo-random number generators in applications that rely on the uniformity of a distribution. Many quasi-random number generators exist such as the Hammersley point set [Hammersley 1960], the Sobol sequence [Sobol' 1967], the Faure sequence [Faure 1982], and the Niederreiter sequence [Niederreiter 1987], which are all examples of digital sequences. The one that concerns us is the generalized Halton sequence, described in the following section.

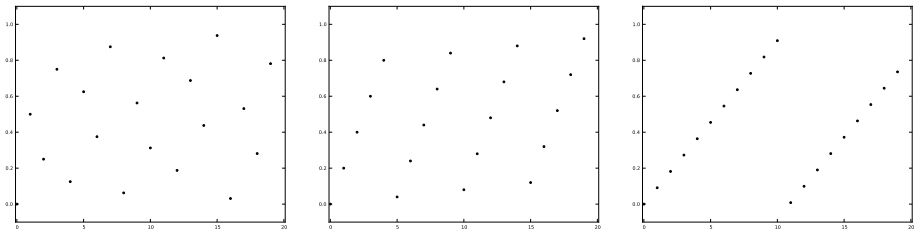
4.1. **Generalized Halton Sequences.** The van der Corput sequence [van der Corput 1935] is a unidimensional low-discrepancy sequence for which the n^{th} element in base b, $x_{n,b} \in [0, 1)$, is given by

$$x_{n,b} = \sum_{i=1}^{k} d_i b^{-i}, \tag{13}$$

where $d_1, \ldots, d_k$ are the digits of the expansion of n in base b,

$$n = \sum_{i=1}^{k} d_i b^{i-1} = d_k d_{k-1} \ldots d_1.$$

The Halton sequence [Halton 1960] is the multidimensional generalization of the van der Corput sequence. It is obtained by grouping multiple van der Corput



(a) Base 2 (b) Base 5 (c) Base 11

FIGURE 2. The first 20 points $(i, x_{i,b})$ of three van der Corput sequences in base b.

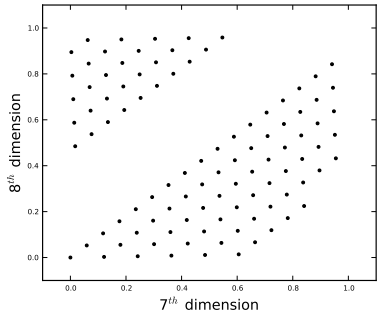


FIGURE 3. First 100 points of the bidimensional projection over coordinates 7 and 8 of the Halton sequence.

sequences with prime bases p_i, where p_i is the i^{th} prime number and has its n^{th} point in s dimensions defined by

$$\mathbf{x}_n = (x_{n,p_1}, \ldots, x_{n,p_s}),$$

where each x_{n,p_i} is given by (13).

The Halton sequence suffers from the correlation between the subsequences with high bases [Braaten and Weller 1979]. Those subsequences are monotonically increasing over a succession of b samples. The greater the base, the longer the monotonic sequence is and the more correlated the subsequences are. Figure 2 shows the monotonically increasing van der Corput sequences, while Figure 3 shows how this correlation appears between the 7^{th} and 8^{th} dimensions of the Halton sequence.

The generalized Halton sequence [Braaten and Weller 1979] addresses this problem by shuffling the extension of each number in its associated base. As in (13), each element is given by

$$x_{n,b}^{\boldsymbol{\pi}} = \sum_{i=1}^{k} \pi_{d_i} b^{-i}, \tag{14}$$

where $\boldsymbol{\pi}$ is a permutation of $[0\ 1\ \cdots\ b-1]$, with $\pi_0 = 0$ so that $\forall n, x_{n,b}^{\boldsymbol{\pi}} \neq 0$. The n^{th} point of the generalized Halton sequence in s dimensions is defined by

$$\mathbf{x}_n^{\boldsymbol{\Pi}} = (x_{n,p_1}^{\boldsymbol{\pi}_1}, \ldots, x_{n,p_s}^{\boldsymbol{\pi}_s}),$$

where each $x_{n,p_i}^{\boldsymbol{\pi}_i}$ is given by (14) and $\boldsymbol{\Pi} = [\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_s]$. Many authors have proposed different permutations producing good-quality low-discrepancy sequences [Atanassov and Durchova 2003; Chi et al. 2005; Faure and Lemieux 2009; Kocis and Whiten 1997; Wang and Hickernell 2000].

4.2. **Linear Digit Scrambling.** The linear digit scrambling is a class of permutations that are frequently used to create the configuration of each dimension without dealing directly with a long permutation vector $\boldsymbol{\pi}$. A permutation is then represented by multipliers f_j instead of a full vector. The full permutation can be retrieved by

$$\pi_j = (f_j j + g_j) \mod b_j, \tag{15}$$

where g_j is called a *digital shift*, both $f_j \neq 0$ and g_j are in $\{0, 1, \ldots, b-1\}$, and f_j and b_j are coprime. In *random* linear digit scrambling, f_j and g_j are chosen uniformly in the allowed interval, for each j and often, $f_j = f$ for all j. The translation added by the digital shift is generally used to randomize copies of a particular sequence in order to obtain an unbiased estimator. More details on linear digit scrambling can be found in Lemieux [2009].

5. Evolving Sequences

This section presents a novel way to configure NOLHs and generalized Halton sequences using an evolutionary algorithm. Latin hypercubes have already been built using evolutionary optimization [Bates et al. 2004; Liefvendahl and Stocki 2006]. Both these methods encode the sampling in the individual directly, while in the present work, the construction algorithm introduced in Section 3 is used. The state-of-the-art for building NOLHs is presented in Cioppa and Lucas [2007] and is achieved through random permutation of the base vector. To our knowledge evolutionary algorithms have not yet been used to evolve any kind of quasi-random sequence generators. Some experiments with heuristic methods have been made to tune the permutations of the Halton sequence. Tuffin [1998] proposes a Monte-Carlo search for a permutation or a group of permutations that minimize a discrepancy measure of the sequence. Faure and Lemieux [2009] suggest to pick from a well-chosen subset of the possible permutations the one that minimizes the discrepancy of the bidimensional projections of the current dimension with several preceding ones. On a different note, Koza [1991] and Sipper and Tomassini [1996] used evolutionary algorithms to generate pseudo-random number generators, the former with genetic programming and the latter evolving cellular automata. We claim that evolving permutations of indices for the NOLH and the generalized Halton sequence should allow an initial population of good sequences, equivalent to solutions obtained by random permutation, to move toward excellent configurations optimizing the permutation according to discrepancy and other measures.

5.1. **Evolutionary Algorithms.** Evolutionary algorithms have been introduced in the early 1970s, by Holland [1975]. They are population-based optimization algorithms inspired by the Darwinian evolution theory. At each generation (step), solutions are varied (usually by crossover and mutation) in order to produce new solutions. The fittest solutions are then selected to reproduce themselves in the next generation allowing good characteristics (building blocks) to be passed over and recombined in the next solutions in order to produce even better solutions.

initialize $P_{\mathrm{p}}^{(1)} \leftarrow [(\mathbf{x}_i, f(\mathbf{x}_i)), i = 1, \ldots, \mu]$
for g **in** $1 \,..\, G$:
  for l **in** $1 \,..\, \lambda$:
    if crossover:
      $F \leftarrow \text{select\_random}\left(P_{\mathrm{p}}^{(g)}, 2\right)$
      $\mathbf{o}_l \leftarrow \text{mate}\,(F)$
    else if mutation:
      $F \leftarrow \text{select\_random}\left(P_{\mathrm{p}}^{(g)}, 1\right)$
      $\mathbf{o}_l \leftarrow \text{mutate}\,(F)$
    end
  end
  $P_{\mathrm{o}}^{(g)} \leftarrow [(\mathbf{o}_l, f(\mathbf{o}_l)), l = 1, \ldots, \lambda]$
  $P_{\mathrm{p}}^{(g+1)} \leftarrow \text{select}\left(\left[P_{\mathrm{p}}^{(g)}\ P_{\mathrm{o}}^{(g)}\right], \mu\right)$
end

FIGURE 4. Simple evolutionary algorithm pseudocode.

In evolutionary computation, the solutions to the problem are called individuals, all solutions present at the same time in the algorithm are part of what is called a population. The individuals are composed of a genotype that is manipulated by the variation operators in the evolutionary algorithm. Sometimes, the genotype is not directly a solution to the problem, but it needs to be developed in order to be a viable solution to the problem; this developed solution is called the phenotype. To each individual is associated a fitness that indicates how well the proposed solution addresses the problem.

The pseudocode for a very simple evolutionary algorithm is presented in Figure 4. The first step is to initialize, at random, a number of individuals $\mathbf{x}_i$ equal to the predefined population size μ and put them in the initial parental population $P_{\mathrm{p}}^{(1)}$. The fitness of each initial solution is given by the fitness function $f(\cdot)$. Then, the evolution loop consists of three phases: the variation, the evaluation, and the selection. First, the variation phase is responsible for producing λ offspring via crossover and mutation. According to the crossover and mutation probabilities, a single operation is chosen per iteration. In case of a crossover, two individuals are selected at random from the parental population of the current generation. Those two individuals are mated to produce a new individual that is put in the offspring array $\mathbf{o}$. In case of a mutation, a single individual is chosen and mutated, and the resulting solution is put in the offspring array. After all offspring have been produced, they are evaluated using the fitness function $f(\cdot)$ and the results are stored in the offspring population $P_{\mathrm{o}}^{(g)}$. The final step is to select the next μ sized parental population $P_{\mathrm{p}}^{(g+1)}$ from the union of both the parental and offspring populations, according to the fitness of the individuals. Finally, the evolution continues until a predefined number of generations are completed. Specific operators for the crossover, mutation, and selection are presented respectively in Table 3.

5.2. **Representation.** Sequences presented in Sections 3 and 4 are both configured using permutations of indices $[a_1\ a_2\ \cdots\ a_n]$, where n is the number of elements needed in the configuration vector, and $a_i \in \{1, 2, \ldots, n\}, a_i \neq a_j, \forall i \neq j$. The

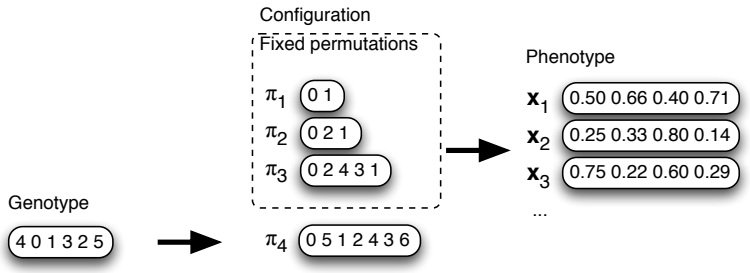


FIGURE 5. Representation of the generalized Halton sequence's genotype.

individuals in the evolved populations are therefore composed of one or more of these fixed-length integer-valued vectors.

The NOLH is directly configured using the permutation of indices as base vector $\mathbf{e}$. The length of the base vector is predetermined from the number of variables needed through (9). The genotype of each individual is then composed of a single permutation vector, while the phenotype is the NOLH matrix described in (12).

The generalized Halton sequence representation is also a permutation vector, which is directly used in (14). One permutation $\boldsymbol{\pi}_i$ is required per dimension, the length of this permutation being determined by the i^{th} prime number p_i. The evolutionary algorithm optimizes a single permutation by evolution. Starting with the first permutation $\boldsymbol{\pi}_1 = [0\ 1]$, dimensions are optimized one at a time extending the configuration $\boldsymbol{\Pi}$ that was found for the first s dimensions by appending to it the new optimized permutation $\boldsymbol{\pi}_{s+1}$. This representation allows a configuration to be built using the same permutations for the n first dimensions for two sequences respectively in n and m dimensions, with $n < m$. Figure 5 shows an example of such a representation when evolving the 4^{th} dimension. The genotype is the current dimension's permutation being optimized, its effective length is $p_4 - 1 = 6$ according to (14). The complete configuration is built by adding the current genotype to the best permutations previously found. Finally, the phenotype is the generalized Halton sequence's N first numbers in 4 dimensions.

5.3. **Variation Operators.** The variation operators used to manipulate the genotypes do respect the representation by vector of indices. The crossover used is the Uniform Partially Matched Crossover (UPMX) (Figure 6(a)) [Cicirello and Smith 2000]. It permutes pairs of values chosen at random positions from the two parents. Moreover, when provided with two valid permutation genotypes, it produces two valid permutation children. In order to fit in the algorithm of Figure 4, only the first child is returned. The chosen mutation is a simple shuffling applied uniformly over some randomly selected chromosomes (see Figure 6(b)). It is called Uniform Partial Shuffling Mutation (UPSM). Every gene of the selected chromosomes has the same probability of being shifted with another one.

5.4. **Performance Measures.**

5.4.1. *Modified L_2 discrepancy ($M_{2,N}(\mathcal{S})^2$).* As described in Section 2, the discrepancy is the measure generally accepted for assessing the nonuniformity of a distribution. The modified L_2 discrepancy of (6) has been chosen for two reasons. First, Fang et al. [2000] stated that it allows the projection's uniformity to be considered

input: Two parents $\mathbf{x}_1, \mathbf{x}_2$ and
a matching probability p
output: Two children $\mathbf{y}_1, \mathbf{y}_2$

initialize children $\mathbf{y}_1 = \mathbf{x}_1, \mathbf{y}_2 = \mathbf{x}_2$
for i **in** $1..n$:
  Draw a random number $0 \leq q \leq 1$
  if $q < p$:
    $a \leftarrow$ Index of $y_{2,i}$ in $\mathbf{y}_1$
    $b \leftarrow$ Index of $y_{1,i}$ in $\mathbf{y}_2$
    Swap values $y_{1,i}$ and $y_{1,a}$
    Swap values $y_{2,i}$ and $y_{2,b}$
end

(a) UPMX

input: One individual $\mathbf{x}$ and
a swapping probability p
output: One mutant $\mathbf{y}$

initialize mutant $\mathbf{y} = \mathbf{x}$
for i **in** $1..n$:
  Draw a random number $0 \leq q \leq 1$
  if $q < p$:
    Draw at random $j \in \{1, \ldots, i-1, i+1, \ldots, n\}$
    Swap values y_i and y_j
end

(b) UPSM

FIGURE 6. Pseudo-code for the operators used in our evolutionary algorithm.

over all subdimensions. Second, it facilitates the comparison of our results for the NOLH with those of Cioppa and Lucas [2007].

5.4.2. *Euclidean maximin distance (*$\mathrm{Mm}(\mathcal{S})$*).* The Euclidean maximin distance is the second measure of uniformity used by Cioppa and Lucas [2007]. It measures the smallest distance between any two points of the distribution. The larger the Euclidean maximin distance is, the more space is covered by the distribution [Johnson et al. 1990]. It is defined as

$$\mathrm{Mm}(\mathcal{S}) = \min(\{d_{1,2}, d_{1,3}, \ldots, d_{s-1,s}\}), \tag{16}$$

where $d_{i,j} = \sqrt{\sum_{k=1}^{s} (\mathcal{S}_{i,k} - \mathcal{S}_{j,k})^2}$.

5.4.3. *Maximum pairwise correlation (*$\mathrm{MPwC}(\mathcal{S})$*).* The maximum pairwise correlation assesses the strongest correlation between two dimensions of the sampling matrix. It is defined as

$$\mathrm{MPwC}(\mathcal{S}) = \max(\{\rho_{1,2}, \rho_{1,3}, \ldots, \rho_{s-1,s}\}), \tag{17}$$

$$\rho_{i,j} = \left| \frac{\sum_{k=1}^{s} (\mathcal{S}_{i,k} - \bar{\mathcal{S}}_i)(\mathcal{S}_{j,k} - \bar{\mathcal{S}}_j)}{\sum_{k=1}^{s} (\mathcal{S}_{i,k} - \bar{\mathcal{S}}_i)^2 \sum_{k=1}^{s} (\mathcal{S}_{j,k} - \bar{\mathcal{S}}_j)^2} \right|,$$

where $\bar{\mathcal{S}}_x$ is the average over all coordinates of the x^{th} point of set $\mathcal{S}$. Cioppa and Lucas [2007] state that the maximum pairwise correlation of a sampling matrix shall not be greater than 0.03 to be nearly orthogonal.

5.4.4. *Condition number (*$\mathrm{Cond}(\mathcal{S})$*).* The condition number is usually used in linear algebra to evaluate the amenability of a problem for numerical computation. Here, it is used in order to compute the orthogonality of the points in the distribution. Cioppa and Lucas [2007] mention that the condition number shall not be greater than 1.13 for a sampling matrix to be nearly orthogonal. It is defined as

$$\mathrm{Cond}(\mathcal{S}) = \frac{\psi_1}{\psi_n}, \tag{18}$$

where ψ_1 and ψ_n are respectively the smallest and largest singular values of the matrix $\mathcal{S}^T\mathcal{S}$.

The performance of the NOLH obtained with configuration $\mathbf{x}$ is given by a combination of the four preceding measures on the produced point set $\mathcal{S}_\mathbf{x}$, (6) and (16) evaluate the uniformity, while (17) and (18) assess the orthogonality. The four independent fitness measures for the NOLH are

$$\begin{aligned} g_{1,\text{NOLH}}(\mathcal{S}_\mathbf{x}) &= M_{2,N}(\mathcal{S}_\mathbf{x})^2, \\ g_{2,\text{NOLH}}(\mathcal{S}_\mathbf{x}) &= \text{Mm}(\mathcal{S}_\mathbf{x}), \\ g_{3,\text{NOLH}}(\mathcal{S}_\mathbf{x}) &= g_{\text{Cond}}(\mathcal{S}_\mathbf{x}), \\ g_{4,\text{NOLH}}(\mathcal{S}_\mathbf{x}) &= g_{\text{Cor}}(\mathcal{S}_\mathbf{x}), \end{aligned}$$

with

$$\begin{aligned} g_{\text{Cond}}(\mathcal{S}_\mathbf{x}) &= \min(1, 0.03/\text{Cond}(\mathcal{S}_\mathbf{x})), \\ g_{\text{Cor}}(\mathcal{S}_\mathbf{x}) &= \min(1, 1.13/\text{MPwC}(\mathcal{S}_\mathbf{x})). \end{aligned}$$

In this fitness measure, the Euclidean distance, the correlation, and the condition number are maximized, while the discrepancy is minimized. The quality of a generalized Halton sequence configuration $\mathbf{x}$ is evaluated solely by the modified L_2 discrepancy over the formed point set $\mathcal{S}_\mathbf{x}$,

$$g_\text{H}(\mathcal{S}_\mathbf{x}) = M_{2,N}(\mathcal{S}_\mathbf{x})^2.$$

In the algorithm of Figure 4, the fitness function $f(\cdot)$ is defined on an individual $\mathbf{x}$, while the measures presented in this section compute the characteristics of the points set $\mathcal{S}_\mathbf{x}$ developed from the configuration $\mathbf{x}$. The fitness function for the NOLH is $f(\mathbf{x}) = g_\text{NOLH}(\mathcal{S}_\mathbf{x})$, with $g_\text{NOLH}(\cdot)$ returning the vector $[g_{i,\text{NOLH}}(\cdot), i = 1, \ldots, 4]$. Section 6.1 presents how these vectors are compared with each other in order to rank the solutions during the selection process. The fitness function for the generalized Halton sequence is directly $f(\mathbf{x}) = g_\text{H}(\mathcal{S}_\mathbf{x})$.

6. Generated Sequences and Their Uniformity

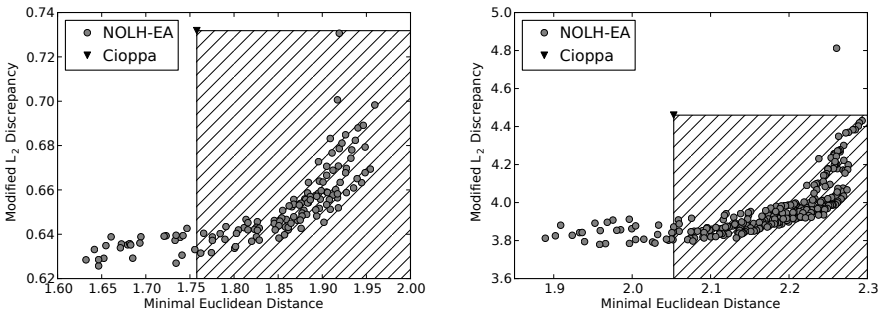
This section presents the experiments on the optimization of low-discrepancy sequences with an evolutionary algorithm. The uniformity of the produced sequences is then compared with respective state-of-the-art relevant methods found in the literature – Cioppa and Lucas' [2007] NOLH, Atanassov and Durchova's [2003], Faure and Lemieux' [2009], and Sobol' [1967] quasi-random sequences.

6.1. NOLH.

Experiments with NOLHs were made using order $m = 5$ and 6. From the chosen order and (7) and (8), the number of samples in the constructed NOLH is $n = 33$ and 65, and their dimensionality is $s = 11$ and 16. Since there are two measures of quality and two near orthogonality criteria composing the fitness of a NOLH, a simple selection scheme cannot be used. Selection is then made by comparing the individuals as per Pareto dominance with the NSGA-II algorithm [Deb et al. 2002]. The concept of Pareto dominance is used in multiobjective optimization to describe a solution that improves at least one of the objectives without deteriorating any. The former solution is then said to Pareto-dominate the latter solution. The Pareto front of a problem is the set of all solutions that are not directly Pareto-dominated by any other solution. Interested readers are referred to Deb [2000] for more details on the use of Pareto dominance in multiobjective optimization. The other parameters related to the evolutionary algorithm are presented in Table 3.

TABLE 3. Evolution parameters for the Optimization of the Low-Discrepancy Sequences

Parameters	NOLH	NOLH	Halton	Halton	Halton
Dimensionality	11	16	2-20	21-50	51-100
Number of samples taken	33	65	2500	2500	2500
Number of generations	500	500	250	500	1000
Population size	1000	1000	500	750	750
Selection type	NSGA-II	NSGA-II	Tourn.	Tourn.	Tourn.
Tournament size	-	-	10	10	10
Crossover type	UPMX	UPMX	UPMX	UPMX	UPMX
Crossover probability	0.5	0.5	0.5	0.5	0.5
Matching probability (UPMX)	0.2	0.2	0.2	0.2	0.2
Mutation type	UPSM	UPSM	UPSM	UPSM	UPSM
Mutation probability	0.1	0.1	0.3	0.3	0.3
Swapping probability (UPSM)	0.05	0.05	0.02	0.02	0.02

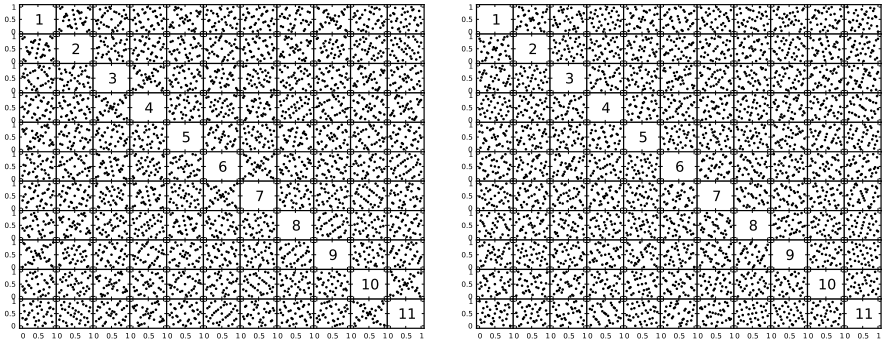


(a) Order 5 (b) Order 6

FIGURE 7. Comparison of the performance of the NOLHs.

Figure 7 shows all solutions present in the 10 Pareto fronts found in 10 independent experiments of evolving NOLHs of order 5 and 6. Each solution presented respects both nearly orthogonal criterion that are the maximum pairwise correlation and the condition number. It is clear that the evolutionary algorithm is able to optimize the base vector in order to produce excellent NOLHs as a large majority of the produced solutions dominates in the Pareto sense Cioppa and Lucas' [2002] best NOLH (hatched zone). In fact, 82% and 90% of the solutions found dominate their corresponding Cioppa and Lucas' solution. In their experiments, Cioppa [2002] builds the best NOLH after examining 1 and 2 million different designs for the order 5 and 6, respectively. The evolutionary algorithm evaluated roughly 500 000 different NOLHs, proving that the EA is able to walk its way through the fitness landscape of possible configurations.

From all produced NOLHs, we found that the ones that were best in optimizing both measures from the 10 Pareto fronts have a modified L_2 discrepancy of 0.6609 and 3.935, and a minimal Euclidean distance of 1.936 and 2.244 for orders 5 and 6, respectively. In comparison, Cioppa and Lucas' best design has a discrepancy of



(a) Cioppa and Lucas' design (b) Evolutionary design

FIGURE 8. Comparison of the distribution between Cioppa and Lucas' and evolutionary NOLH of order 5. Each cell represents a two-dimensional projection of the point set for the dimensions given on the diagonal.

0.7318 and 4.46, and a minimal Euclidean distance of 1.758 and 2.053 each. The evolutionary algorithm is then able to decrease the discrepancy by 9.6% and 11.7%, and increase the minimal Euclidean distance by 10.1% and 9.3%. Figure 8 presents a pairwise projection for a visual comparison of Cioppa and Lucas' solution with the best evolutionary design. In the first one, obvious poor space-filling patterns are present between dimensions 3 and 4, 6 and 7, 10 and 11, and 1 and 9, by the observation of the visible "X". No such pattern exists in the evolutionary NOLH of Figure 8(b). The electronic appendix contains the presented NOLHs' base vector along with those of order 7 and 8.

6.2. **Generalized Halton Sequence.** The optimization of a configuration for the generalized Halton sequence has been carried out for sequences of dimensionality up to 100. In opposition to the NOLH, an infinite number of points may be drawn from this generator, then a fixed number of points has to be chosen for the evaluation. This number of samples must be large enough to enable a good estimation of the uniformity of the sequence. Faure and Lemieux [2009] state that in order to obtain a good estimate of the uniformity of the bidimensional projections of dimensions i and j, $n = p_i p_j$ samples shall be drawn, where p_m is the m^{th} prime number. Since the 100^{th} prime number is 541, it would require evaluating the discrepancy of a sequence of about 300 000 samples, which is not tractable for an algorithm such as the modified L_2 discrepancy given its complexity of $O(n^2 s)$. Faure and Lemieux [2009] again mention that $n = 2\,500$ is a good compromise to allow the search to be made in a reasonable time; they search for a suitable sequence of up to 360 dimensions and $p_{360} = 2\,423$. The samples evaluated are then the first 2 500 points of the generalized Halton sequence configured by the individuals from the evolutionary algorithm. The remaining evolutionary parameters are presented in Table 3.

Tables 4 and 5 present the discrepancy of some well-known low-discrepancy sequences compared to the discrepancy of the generalized Halton sequences configured

TABLE 4. Modified L_2 Discrepancy of Some Well-Known Sequences for the First 2 500 Points

Sequence	$s = 20$	$s = 50$	$s = 100$
Evolutionary Algorithm	0.4166	172851	0.2604×10^{14}
Random	0.4850	213627	1.140×10^{14}
Atanassov-Durchova	0.4788	247088	1.771×10^{14}
Faure-Lemieux	0.4783	258004	1.619×10^{14}
Halton	1.469	4.8×10^{8}	3.947×10^{23}
Sobol	0.4882	222586	1.506×10^{14}

TABLE 5. L_2-Star Discrepancy of Some Well-Known Sequences for the First 2 500 Points.

Sequence	$s = 20$	$s = 50$	$s = 100$
Evolutionary Algorithm	0.3550×10^{-9}	2.359×10^{-21}	0.7772×10^{-45}
Random	0.1790×10^{-9}	3.782×10^{-21}	0.4799×10^{-39}
Atanassov-Durchova	0.4000×10^{-9}	344.2×10^{-21}	29.67×10^{-36}
Faure-Lemieux	0.4718×10^{-9}	816.5×10^{-21}	1.043×10^{-36}
Halton	4.061×10^{-9}	2.734×10^{-8}	2.205×10^{-8}
Sobol	0.3525×10^{-9}	36.92×10^{-21}	1.034×10^{-36}

by the Evolutionary Algorithm (EA). The random configuration presents the minimal discrepancy found among 1 million random configurations. According only to the discrepancy measures, it can be seen that, in comparison to a random walk, the evolutionary algorithm offers a substantial improvement in the optimization of the configuration of the sequences. Moreover, the configuration of Atanassov and Durchova (AD) and Faure and Lemieux (FL) have also been made according to a discrepancy criterion. The former makes use of *admissible integers* described in Atanassov [2004] for which "an estimate of [the sequences'] discrepancy with a very small leading term can be obtained" [Atanassov and Durchova 2003]. The latter, for its part, directly optimizes the discrepancy of the sequence following a recursive method similar to ours. Briefly, Faure and Lemieux [2009] ranked each linear scrambling multiplier according to an L_2 discrepancy bound given in Faure [2006] and, from the best 32 multipliers in each dimension, they chose the one that minimized the bidimensional discrepancy of the first 2 500 points over the 7 preceding dimensions. Our sequence shows a modified L_2 and a L_2-star discrepancy measure well below those sequences, showing once again the power of the evolutionary algorithm to optimize each permutation according only to the discrepancy criterion.

7. Testing the Generalized Halton Sequence Configuration

In addition to the uniformity of the produced sequence, there is a large battery of tests available for assessing the quality of a low-discrepancy sequence. These tests are numerical estimations of complex integrands. This section presents the efficiency of the optimized sequence EA on estimating those integrands.

7.1. **Effective Dimension.** The effective dimension is a measure over a function to assess how difficult it is to integrate with numerical methods. There are two

effective dimension notions that are suited to different kind of functions. The effective dimension in the *truncation sense* reflects the number of important variables that are required to estimate the integrand. For example, if a function has an effective dimension in the truncation sense d_T, then it can be fairly approximated by another function that depends only on d_T variables. On the other hand, the effective dimension in the *superposition sense* is used when all variables of the function are equally (or almost equally) important. It measures the effect of the interactions between those variables. Concretely, if the effective dimension in the superposition sense of a function is d_S, a sequence with good d_S-dimensional projections for all bases should be able to provide a good estimate of the function. More details on the effective dimension can be found in Caflisch et al. [1997] and Wang and Fang [2003].

7.2. **Test Functions.** Many test functions have been used in the literature in order to compare the quality of different low-discrepancy sequences. We chose three that cover the two notions of effective dimension of the preceding section. We have

$$f_1(\mathbf{x}) = \prod_{i=1}^{s} \frac{|4x_i - 2| + a_i}{1 + a_i}, \tag{19}$$

$$f_2(\mathbf{x}) = \prod_{i=1}^{s} 1 + c(x_i - 0.5), \tag{20}$$

$$f_3(\mathbf{x}) = \alpha_s \pi^{s/2} \cos\left(\sqrt{\frac{1}{2}\sum_{i=1}^{s} [\Phi^{-1}(x_i)]^2}\right), \tag{21}$$

where Φ^{-1} is the inverse standard normal cumulative distribution function.

The first function (19) is used by Atanassov and Durchova [2003] and Faure and Lemieux [2009]. The former use (i) $a_i = 0$, and the latter (ii) $a_i = 0.01$, (iii) $a_i = 1$, (iv) $a_i = i$, (v) $a_i = i^2$, and (vi) $a_i = (s - i + 1)^2$. We tested our sequence on the six proposed choices of a_i and on three different dimensions, $s \in \{20, 50, 100\}$. The effective dimension of this function can be calculated in the truncation sense; it should decrease going from (i) to (v), so that (i) is more difficult to estimate than (v) [Faure and Lemieux 2009]. Case (vi) has the same effective dimension as (v), but, as opposed to the other cases, the last variables are more important than the first ones, that is, the most important variables are in order, $x_s, x_{s-1}, \ldots, x_1$. This function integrates to 1.

The second function (20) is used by Sobol' and Asostsky [2003]. In f_2, the parameter c is used to adjust the effective dimension in the superposition sense. We used a single combination of parameters for this function, which have been proposed by Sobol' and Asostsky [2003], that is $c = 0.25$ and $s = 96$. The effective dimension in the superposition sense has been computed by Faure and Lemieux [2009] and is 6, using a threshold of 0.99 in the definition of the effective dimension in the superposition sense given in Caflisch et al. [1997]. This function integrates to 1.

The last function (21) is used by Papageorgiou and Traub [1997]. Contrary to f_1 and f_2, it is defined as a summation and is part of a different family called isotropic integrals. The effective dimension in the superposition sense for $s = 20$ is about 3, as computed by Owen [2003]. We tested this function with $s \in \{9, 25, 60, 100\}$

TABLE 6. Estimated Values for the Asian Call Option

s	$K = 45$	$K = 50$	$K = 55$
40	7.0471756201	4.0521833136	2.1055268449
75	7.0147850502	4.0155665867	2.0733021534

dimensions. The α_s parameter has been numerically determined in Papageorgiou and Traub [1997] so that this third function also integrates to 1.

7.3. **Integrand from Finance.** In addition to the three preceding test functions, we tested our sequence on a well-known finance problem, the pricing of an *Asian call option*. This experiment consists in estimating the value C_0 of an option at time 0 using

$$C_0 = E\left[e^{-rU}\max\left(\frac{1}{s}\sum_{i=1}^{s} S(u_i) - K, 0\right)\right]. \tag{22}$$

In (22), U is the expiration time of the contract, K is the strike price, $S(u_i)$ is the price of the asset at s observation times u_i with $0 < u_1 < \ldots < u_s = U$, and r is the risk-free appreciation rate. If we assume the price process follows the Black-Scholes model [Glasserman 2004], the value C_0 from (22) can be rewritten as the integral

$$C_0 = e^{-rU}\int_{[0,1)^s} \max\left(\frac{1}{s}\sum_{i=1}^{s} S(0)e^{\frac{r-\sigma^2}{2}u_i + \sigma\sum_{l=1}^{i}\sqrt{\Delta_l}\Phi^{-1}(x_l)} - K, 0\right) dx_1 \ldots dx_s,$$

where $\Delta_l = u_l - u_{l-1}$. Three test cases for two different dimensionalities, corresponding to those studied by Faure and Lemieux [2009], have been studied, $s \in \{40, 75\}$ and $K \in \{45, 50, 55\}$. The other parameters have been fixed to $S(0) = 50$, $U = 1$ year, $r = 0.05$, $\sigma = 0.3$, and $u_j = j/s$. The exact value of this function is unknown and has been estimated using 20 digital shifts of the 2^{23} first numbers of the Sobol' sequence. Table 6 presents the estimated values found for the Asian call option for the different configurations.

7.4. **Results.** We tested our sequence on the 29 function cases mentioned before, two figures are produced for each function: the absolute error E_N as given in (4), and the variance of the absolute error on 25 independent randomizations of each sequence. For each function, we used 2 000 to 1 000 000 points with an increment of 2 000. Each randomization consists in a digital shift of the original sequence with $\mathbf{v} = [v_1 \ \cdots \ v_s]$ in $[0,1)^s$ so that the i^{th} point $\tilde{\mathbf{x}}_i$ is given by

$$v_j = \sum_{k=1}^{\infty} u_{j,k} b_j^{-k}, \quad x_{i,j} = \sum_{k=1}^{\infty} d_{i,j,k} b_j^{-k},$$
$$\mathbf{x}_i \oplus \mathbf{v} = [\tilde{x}_{i,1} \ \cdots \ \tilde{x}_{i,s}] = \tilde{\mathbf{x}}_i,$$

with

$$\tilde{x}_{i,j} = \sum_{k=1}^{\infty}((u_{j,k} + d_{i,j,k}) \mod b_j) b_j^{-k},$$

as described in L'Ecuyer and Lemieux [2002]. To compare our results, for each case, we also plotted the estimation error of the sequence from Atanassov and Durchova [2003] (AD), Faure and Lemieux [2009] (FL), and Sobol' [1967] (S). The Sobol sequence is configured with the direction numbers given in Joe and Kuo [2003].

The two former sequences have been shown dominant over the standard Monte-Carlo technique, the original Halton sequence, as well as some other generalized Halton configurations [Vandewoestyne and Cools 2006; Wang and Hickernell 2000] in Faure and Lemieux [2009]. The complete list of all figures can be found in the electronic appendix. We present in Figures 9 to 16 the results we consider the most representative. In those figures, the data have been convolved with a unit area rectangle window of width 11 in order to remove the high-frequency noise. This gives a better overview of the tendencies shown in each figure.

The two first cases for function f_1 are very difficult to estimate in all three tested dimensions because of their high effective dimension in the truncation sense. While the integration of the functions in 20 dimensions is still possible and gives comparable results for AD, EA, FL, and S in the deterministic case, using digital shifts with a high number of points reveals a slight edge in favor of EA that is most of the time among the two best sequences as shown in Figures 9 and 10. Figure 11 shows an interesting behavior of the deterministic sequences with a large number of points. Whereas the FL sequence shows a very low integration error for the first section of the figure, the EA and S sequences appear more efficient with a higher number of samples. The subsequent cases for function f_1 are much easier to estimate and the difference between the diverse sequences is less obvious. Figure 12 shows how in 100 dimensions with $a_i = i^2$ the results are very close between all sequences with a minor advantage to EA. The last test case of function f_1 is very interesting, as reversing the dimension order reveals that the generated sequence struggles in the last dimensions compared to how it behaves in the first ones. This can be seen in Figure 13. We discuss this in the next section.

The effective dimension in the truncation sense of functions f_2 and f_3 is smaller than their dimensionality. Nevertheless, these functions are still quite challenging since their effective dimension in the superposition sense is not negligible. Once again, the EA sequence performs as well as the best Halton configuration available in the literature. Figure 14 shows the absolute estimation error of the deterministic sequences on function f_3 in 60 dimensions. It is shown that AD, FL, and EA are all clearly superior to S for the deterministic estimation. However, it is noticed that the variances for the randomized case of this function are similar for all sequences.

For the last function used, the Asian call option, there is almost no distinction between the four tested sequences with $K \in \{45, 50, 55\}$ and $s \in \{40, 75\}$. Figures 15 and 16 show respectively the absolute error and estimated variance for the case $K = 45$ and $s = 40$. Even tough the deterministic EA sequence appears to be disadvantaged on this problem, its variance is comparable with the other sequences.

7.5. **Discussion.** The main point that should be noted here is that the evolved sequence is competitive with the best sequences from the literature. Those sequences have been proved superior, in every tested function, to the simple Monte-Carlo sampling, the original Halton sequence, and some configurations of the Halton sequence present in the literature.

In our opinion, the explanation of the poor results for function f_1 with the reversed dimension importance is twofold. First, the manner in which the configuration is built with the evolutionary algorithm, that is to fix consecutively each one-dimensional sequence after another to generate the final s-dimensional configuration, introduces a bias in the search for the next dimension permutations that is dependent on the previous ones. On the same line, the choice of measuring the

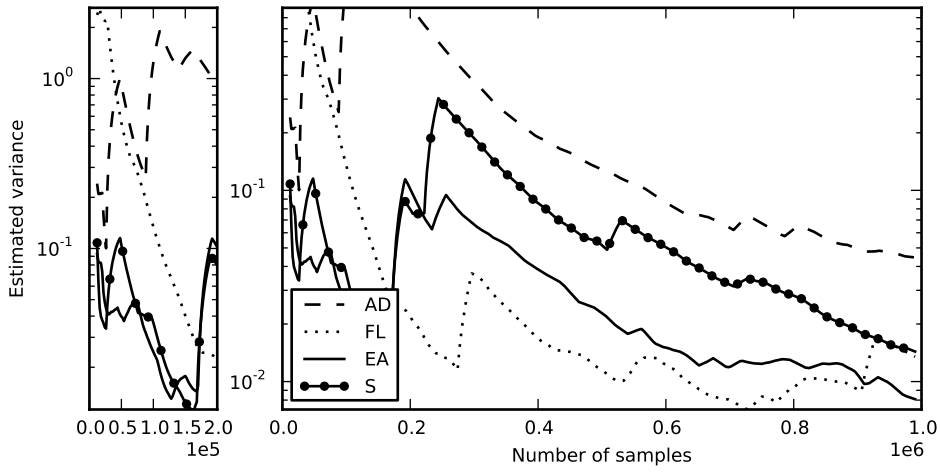


FIGURE 9. Variance of the absolute error on test function f_1, with $a_i = 0.01$ and $s = 50$.

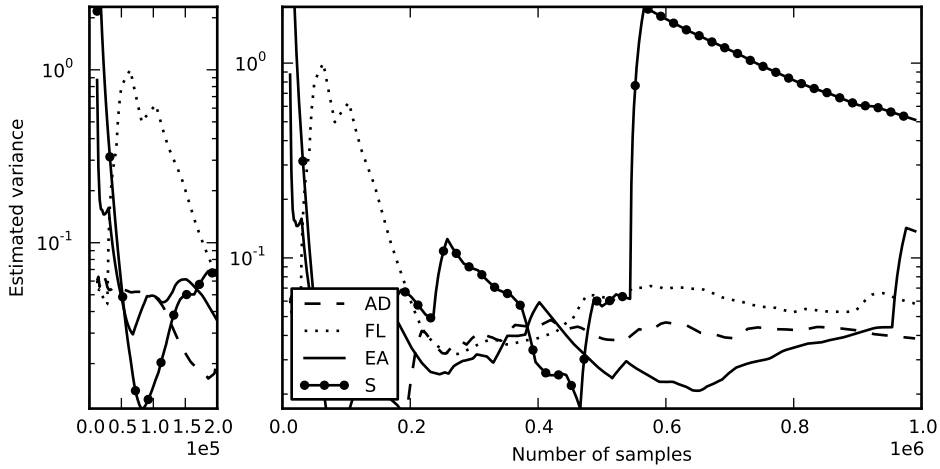


FIGURE 10. Variance of the absolute error on test function f_1, with $a_i = 0.01$ and $s = 100$.

discrepancy on dimensions 1 to s at each step may have prevented the optimizer from finding good permutations that would have given lower discrepancy measures between *local* dimensions. Locality between dimensions is defined freely here as dimensions that are close together in term of their index. For example, dimensions 3 and 5 are closer than dimensions 2 and 28. Knowing that the effective dimension should be equivalent for function f_1 in cases (v) and (vi), the uniformity in all dimensions (including all n-dimensional projections, with $n = 1, \ldots, s$) is not mandatory to approximate the function. Thus, the discrepancy measure could have been restricted to more local dimensions, say $s - w$ to s, to allow more flexibility in the search of good permutations. However, this setting would come with a loss of generality for the produced configuration on problems that does not respect this

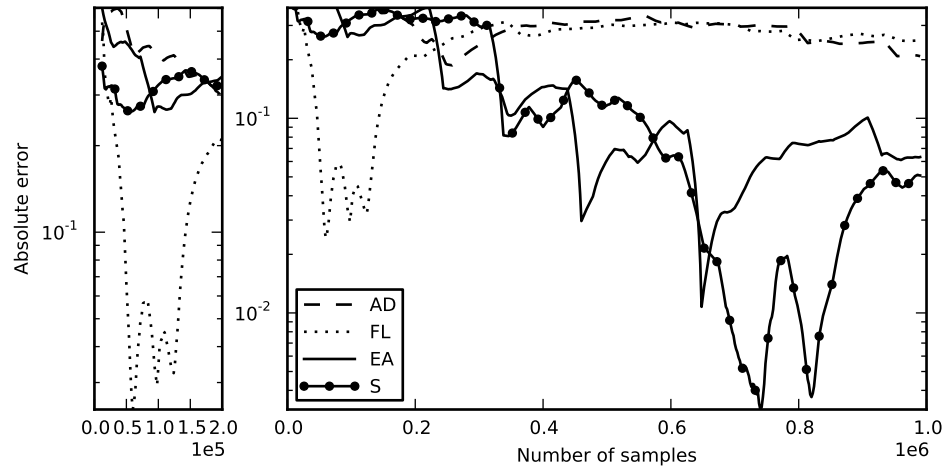


FIGURE 11. Absolute error of the deterministic sequences on test function f_1, with $a_i = 0.01$ and $s = 50$.

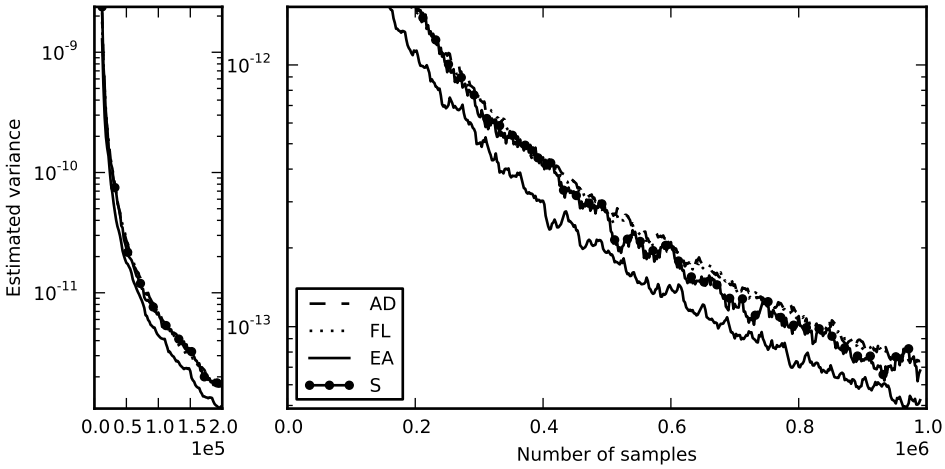


FIGURE 12. Variance of the absolute error on test function f_1, with $a_i = i^2$ and $s = 100$.

last assertion. Of course, adding prior knowledge in the black-box optimizer would facilitate the task of finding specialized configuration for specific problems. But, our goal being to provide a novel method to optimize the configuration of low-discrepancy number generators, the more general setting has been used to show its efficiency on a great number of cases.

Second, the computational effort used to optimize the higher dimensions is proportionally less than for the optimization of the lower dimensions. As the length of the permutation grows, the number of generations and the population size of the evolutionary algorithm are kept constant over a large number of dimensions.

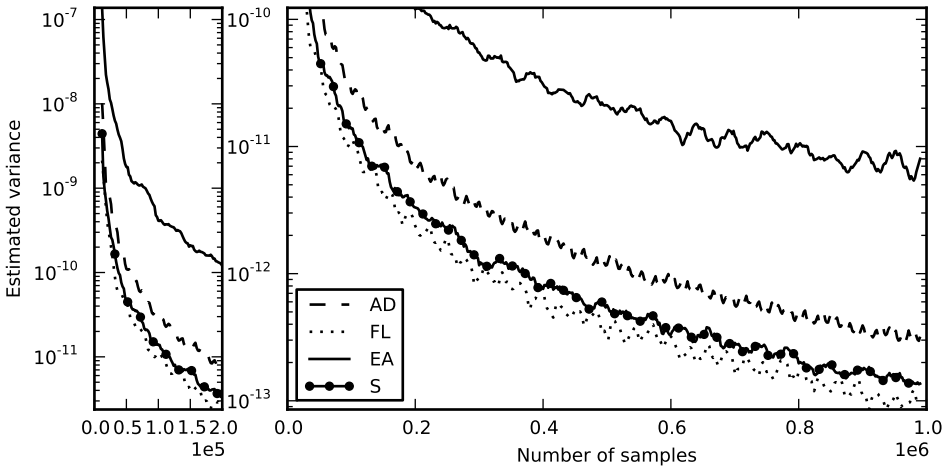


FIGURE 13. Variance of the absolute error on test function f_1, with $a_i = (s - i + 1)^2$ and $s = 50$.

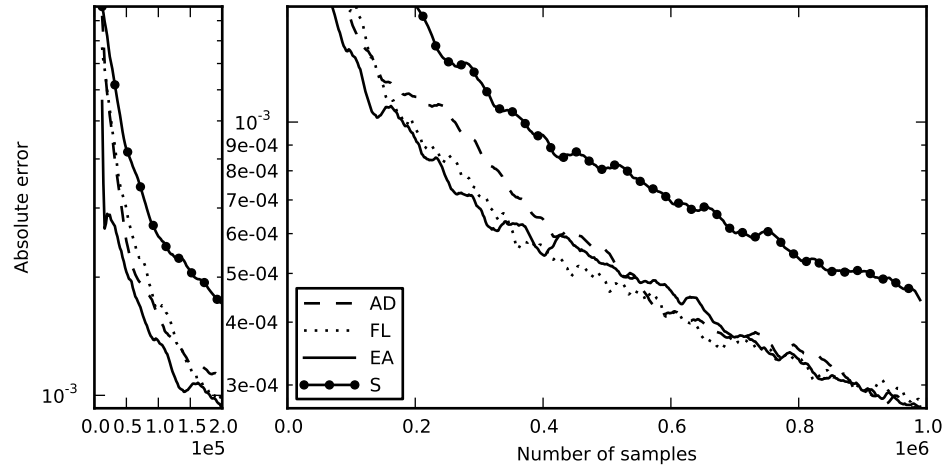


FIGURE 14. Absolute error of the deterministic sequences on test function f_3, with $s = 60$.

Traditional discrepancy measures can be somewhat disappointing for sequences in high dimensions as they ignore the lower-order projections [Wang and Sloan 2008]. Multiple other discrepancy measures exist that take into account the quality of selected projections of the point set in lower-dimensional spaces, for example, the weighted L_2-star discrepancy [Hickernell 1998; Sloan and Woźniakowski 1998] or the order-l discrepancy [Wang and Sloan 2008].

The proposed generalized Halton sequence employs full general permutations of p elements in base p. This kind of permutation is unfavored compared to the more classical use of well-chosen multipliers presented in Atanassov and Durchova [2003], Chi et al. [2005], Faure and Lemieux [2009], and Kocis and Whiten [1997]. In fact,

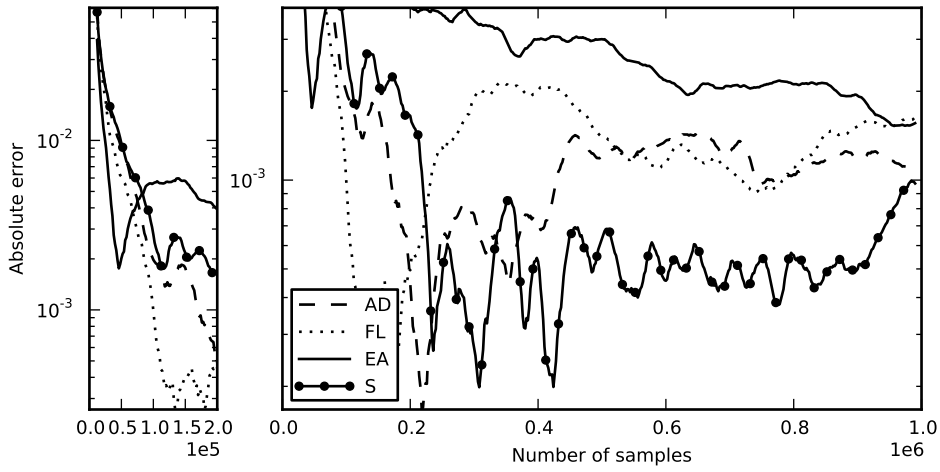


FIGURE 15. Absolute error of the deterministic sequences on the Asian call option, with $K = 45$ and $s = 40$.

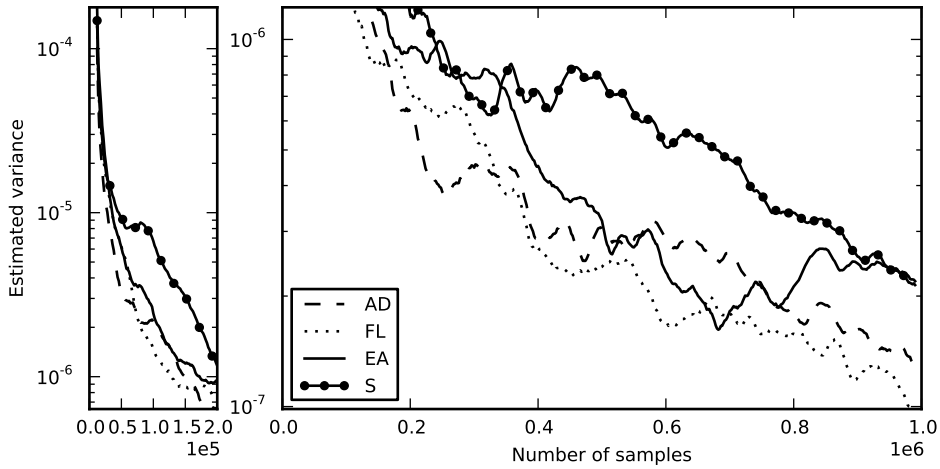


FIGURE 16. Variance of the absolute error on the Asian call option, with $K = 45$ and $s = 40$.

the method presented in this article requires p_i integers per dimension, where p_i is the i^{th} prime number, while the uses of multipliers limits the number of integers to 1 per dimension. This leads to an explosion of the number of integers to store when the number of dimensions is high. For example, in 500 dimensions, 842 670 integers are required. Figure 17 shows the number of integers needed by each method as the dimensionality increases.

8. Conclusion

This article shows that optimization algorithms, specifically evolutionary algorithms, can be successfully used in the optimization of low-discrepancy sequences

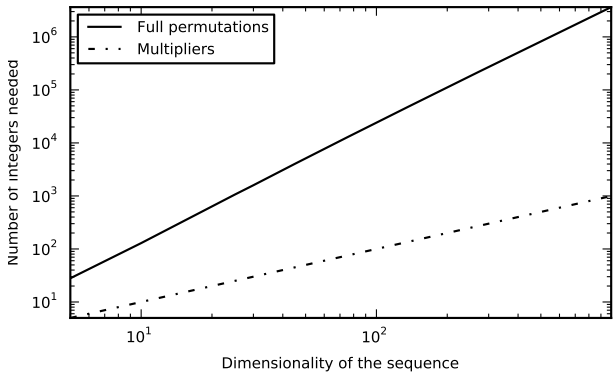


FIGURE 17. Growth of the number of integers to store for the full generalized permutations and multipliers methods.

based only on a discrepancy measure. It also proposes that the produced sequences are competitive against the most recent sequences proposed in the literature for a wide range of functions and dimensions. Therefore, we claim that evolutionary algorithms shall be considered when configuring low-discrepancy sequences, as this method is able to walk its way through the fitness landscape of the problem, to generate efficient quasi-random number generators.

The current work can be expanded in several ways. First, the optimization of low-discrepancy sequence generators is guided by the modified L_2 discrepancy, which represents a specific way of evaluating the space-filling properties from a given number of samples. There may be different measures that lead to better accuracy, or lower complexity, allowing computing discrepancy with more samples, to obtain better measures for a general quasi-random number generator. Thanks to the black-box optimization nature of evolutionary algorithm, if such a measure is identified, the proposed approach for optimizing the NOLH and the generalized Halton generators can still be applied simply by replacing the current discrepancy measure with the new one. This approach would also by applicable for producing a number generator that is designed for a specific purpose, that is if a domain specific performance measure exists it could be used to evolve the configuration of a low-discrepancy number generator with the desired properties.

The permutations found by the evolutionary algorithm for NOLH of order 5 to 8 and for the generalized Halton sequence up to 100 dimensions are available at http://vision.gel.ulaval.ca/~fmdrainville/permutations.html. A simple generator to produce the generalized Halton sequence with custom configurations in C++ and Python is also made available.

ACKNOWLEDGMENTS

This article greatly benefited from the comments of the Editor-in-Chief and Associate Editor. The authors are also grateful to A. Schwerdtfeger for proofreading this manuscript. This work is supported by Defence Research and Development Canada Valcartier (DRDC Valcartier), FQRNT (Québec), and NSERC (Canada). This work has been made possible through the access to the supercomputing facilities of CLUMEQ/Compute Canada.

References

E. I. Atanassov. On the discrepancy of the Halton sequences. *Mathematica Balkanica*, 18:15–32, 2004.

E. I. Atanassov and M. K. Durchova. Generating and testing the modified Halton sequences. In *Proceedings of the Conference on Numerical Methods and Applications (NMA'02)*, pages 91–98, Berlin, 2003. Springer.

A. Auger, M. Jebalia, and O. Teytaud. Algorithms (X, sigma, eta): Quasi-random mutations for evolution strategies. In *Proceedings of the Artificial Evolution Conference (EA'05)*, pages 296–307, Berlin, 2006. Springer.

S.J. Bates, J. Sienz, and V.V. Toropov. Formulation of the optimal Latin hypercube design of experiments using a permutation genetic algorithm. In *Proceedings of the 5th ASMO-UK/ISSMO Conference on Engineering Design Optimization*, 2004.

E. Braaten and G. Weller. An improved low-discrepancy sequence for multidimensional quasi-Monte Carlo integration. *J. of Comput. Phys.*, 33(2):249–258, 1979.

R. E. Caflisch, W. J. Morokoff, and A. B. Owen. Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension. *J. of Comput. Finance*, 1(1):27–46, 1997.

C. Cervellera and M. Muselli. Deterministic design for neural network learning: An approach based on discrepancy. *IEEE Trans. on Neural Netw.*, 15(3):533–544, 2004.

H. Chi, M. Mascagni, and T. T. Warnock. On the optimal Halton sequence. *ACM Trans. Model. Comput. Simul.*, 70(1):9–21, 2005.

V. A. Cicirello and S. F. Smith. Modeling GA performance for control parameter optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 235–242, 2000.

T. M. Cioppa. *Efficient Nearly Orthogonal and Space-Filling Experimental for High-Dimensional Complex Models*. PhD thesis, Naval Postgraduate School, Monterey, CA, September 2002.

T. M. Cioppa and T. W. Lucas. Efficient nearly orthogonal and space-filling Latin hypercubes. *Technometrics*, 49(1):45–55, 2007.

K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK, 2000.

K. Deb, A. Pratab, S. Agarwal, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *IEEE Trans. Evolut. Comput.*, 6(2):849–858, 2002.

K. T. Fang, D. K. J. Lin, P. Winker, and Y. Zhang. Uniform design: Theory and application. *Technometrics*, 42(3):237–248, 2000.

H. Faure. Discrépance de suites associées à un système de numération (en dimension s). *Acta Arithmetica*, 41:337–351, 1982.

H. Faure. Selection criteria for (random) generation of digital (0,s)-sequences. In H. Niederreiter and D. Talay, editors, *Proceedings of the Conference on Monte Carlo Methods and Quasi-Monte Methods*, pages 113–126, Springer-Verlag, Berlin, 2006.

H. Faure and C. Lemieux. Generalized Halton sequences in 2008: A comparative study. *ACM Trans. Model. Comput. Simul.*, 19(4):1–43, 2009. ISSN 1049-3301.

A. Florian. An efficient sampling scheme : Updated Latin hypercube sampling. *Probabil. Engin. Mechan.*, 7(2):123–130, 1992.

P. Glasserman. *Monte Carlo Methods in Financial Enginerring*. Springer-Verlag, New York, 2004.

J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, 1960.

J. M. Hammersley. Monte Carlo methods for solving multivariable problems. *Ann. NY Acad. Sci.*, 86(3):844–874, 1960.

F. J. Hickernell. A generalized discrepancy and quadrature error bound. *Math. Comput.*, 67(221):299–322, 1998. URL citeseer.ist.psu.edu/hickernell97generalized.html.

J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.

S. Joe and F. Y. Kuo. Remark on algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Trans. Math. Softw.*, 29:49–57, 2003.

M. E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *J. Statist. Plan. Infer.*, 26:131–148, 1990.

L. Kocis and W. J. Whiten. Computational investigations of low-discrepancy sequences. *ACM Trans. Math. Softw.*, 23(2):266–294, 1997.

J. R. Koza. Evolving a computer program to generate random numbers using the genetic programming paradigm. In *Proceedings of the 4th International Conference on Genetic Algorithm*, pages 37–44. Morgan Kaufmann, 1991.

P. L'Ecuyer. Quasi-Monte Carlo methods with applications in finance. *Finance Stochast.*, 13:307–349, 2009.

P. L'Ecuyer and C. Lemieux. Variance reduction via lattice rules. *Manag. Sci.*, 46 (9):1214–1235, 2000.

P. L'Ecuyer and C. Lemieux. Recent advances in randomized quasi-Monte Carlo methods. In M. Dror, P. L'Ecuyer, and F. Szidarovszki, editors, *Proceedings of the Conference on Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications*, chapter 20, pages 479–474. Kluwer Academic Publishers, 2002.

C. Lemieux. *Monte Carlo and Quasi Monte Carlo Sampling*. Springer, New York, 2009.

M. Liefvendahl and R. Stocki. A study on algorithms for optimization of Latin hypercubes. *J. Statist. Plan. Infer.*, 136(9):3231–3247, 2006.

M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

W. J. Morokoff and R. E. Caflisch. Quasi-random sequences and their discrepancies. *SIAM J. Sci. Comput.*, 15(6):1251–1279, 1994.

H. Niederreiter. Quasi-Monte Carlo methods and pseudo-random numbers. *Bull. Amer. Math. Soc.*, 84(6):957–1041, 1978.

H. Niederreiter. Point sets and sequences with small discrepancy. *Monatshefte für Math.*, 104:273–337, 1987.

H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, Philadelphia, PA, 1992.

A. B. Owen. The dimension distribution and quadrature test functions. *Statistica Sinica*, 13:1–17, 2003.

A. Papageorgiou and J. F. Traub. Faster evaluation of multidimensional integrals. *Comput. Phys.*, 11(6):574–579, 1997.

M. Sipper and M. Tomassini. Co-evolving parallel random number generators. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the Conference on Parallel Problem Solving from Nature (PPSNIV)*, volume 1141 of *Lecture Notes in Computer Science*, pages 950–959. Springer, 1996.

I. H. Sloan and H. Woźniakowski. When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? *J. Complex.*, 14(1):1–33, 1998.

I. M. Sobol'. The distribution of points in a cube and the approximate evaluation of integrals. *USSR Comput. Math. Math. Phys.*, 7(4):86–112, 1967.

I. M. Sobol' and D. I. Asostsky. One more experiment on estimating high-dimensional integrals by quasi-Monte Carlo methods. *Math. Comput. Simul.*, 62(3-6):255–263, 2003.

B. Tuffin. A new permutation choice in halton sequences. In P. Hellekalek, G. Larcher, H. Niederreiter, and P. Zinterhof, editors, *Proceedings of the Conference on Monte Carlo and Quasi-Monte Carlo Methods*, volume 127 of *Lecture Notes in Statistics*, pages 427–435, Berlin, 1998. Springer.

J. G. van der Corput. Verteilungsfunktionen. In *Akademie van Wetenschappen*, volume 38, pages 813–821, Amsterdam, 1935. KNAW.

B. Vandewoestyne and R. Cools. Good permutations for deterministic scrambled Halton sequences in terms of L_2-discrepancy. *Comput. Appl. Math.*, 189(1,2): 341:361, 2006.

X. Wang and K. T. Fang. The effective dimension and quasi-Monte Carlo integration. *J. Complex.*, 19:101–124, 2003.

X. Wang and F. J. Hickernell. Randomized Halton sequences. *Math. Comput. Model.*, 32(7):887–899, 2000.

X Wang and I Sloan. Low discrepancy sequences in high dimensions: How well are their projections distributed? *J. Comput. Appl. Math.*, 213(2):366–386, 2008.

T. T. Warnock. Computational investigation of low discrepancy point sets. In *Applications of Number Theory to Numerical Analysis*, pages 319–343, New York, 1972. Academic Press.

K. Q. Ye. Orthogonal column Latin hypercubes and their application in computer experiments. *J. Amer. Statist. Assoc.*, 93:1430–1439, 1998.

Appendix A. Permutation for the Nearly Orthogonal Latin Hypercube

Table 7 presents the best configuration found for the nearly orthogonal Latin hypercubes (NOLHs) of order 5 to 8 through an evolutionary optimization. Table 8 contains the columns to remove from the complete designs to get NOLHs for all numbers of factors between 8 and 28. Ready NOLHs with those configurations are available at http://vision.gel.ulaval.ca/~fmdrainville/permutations.html.

Table 7. NOLH's base vector for order 5 to 7.

Order	Base vector $\mathbf{e}$	$M_{2,N}(\mathcal{S})$	EMm($\mathcal{S}$)	MPwC($\mathcal{S}$)	Cond($\mathcal{S}$)
5	[4 14 1 2 16 13 5 8 12 9 6 7 11 3 15 10]	0.6609	1.9355	0.02741	1.1166
6	[5 13 19 23 28 10 12 32 17 2 30 15 6 31 21 8 24 29 9 14 11 22 18 25 3 1 20 7 27 16 26 4]	3.9510	2.2306	0.01827	1.0935
7	[7 8 51 3 40 44 29 19 61 43 26 48 20 52 4 49 2 57 31 30 24 23 56 50 18 59 63 37 38 21 54 9 46 27 36 1 10 42 13 55 15 25 22 45 41 39 53 34 6 5 32 58 16 28 64 14 47 33 12 35 62 17 11 60]	33.464	2.5630	0.00693	1.0413
8	[9 108 39 107 62 86 110 119 46 43 103 71 123 91 10 13 126 63 83 47 100 54 23 16 124 45 27 4 93 74 76 90 30 81 77 53 116 49 104 67 70 82 26 118 55 79 32 109 57 31 22 101 44 87 121 7 37 56 89 115 25 92 85 20 58 52 3 11 106 17 117 38 78 28 59 96 18 97 50 114 112 60 84 1 12 61 98 128 14 42 64 105 68 75 111 34 141 65 99 2 19 33 35 94 51 122 127 36 125 80 73 8 24 21 88 48 69 66 40 15 29 113 72 5 95 120 6 102]	386.80	2.8008	0.00390	1.0197

Appendix B. 20 First Permutations for the Generalized Halton Sequence

Table 9 presents the first 20 permutations used to configure the first 20 dimensions. The remaining permutations are available at http://vision.gel.ulaval.ca/~fmdrainville/permutations.html.

TABLE 8. Columns to remove in respective NOLH to get a designs of lower dimensionality.

Order	Dim.	Column(s) to remove	$M_{2,N}(\mathcal{S})$	EMm($\mathcal{S}$)	MPwC($\mathcal{S}$)	Cond($\mathcal{S}$)
5	8	$\{1,3,10\}$	0.1177	1.1809	0.02741	0.0
	9	$\{6,10\}$	0.2778	1.4170	0.02741	0.0
	10	$\{10\}$	0.3827	1.6654	0.02741	0.0
6	12	$\{2,4,5,11\}$	0.5003	1.3324	0.01827	0.0
	13	$\{3,6,14\}$	0.8493	1.6837	0.01827	0.0
	14	$\{4,5\}$	1.4217	1.6910	0.01827	0.0
	15	$\{6\}$	2.3779	1.7391	0.01827	0.0
7	17	$\{8,11,12,14,17\}$	3.2348	1.8665	0.00693	0.0
	18	$\{8,11,12,17\}$	5.2220	2.1461	0.00693	0.0
	19	$\{10,15,22\}$	8.3445	2.2445	0.00693	0.0
	20	$\{8,12\}$	13.263	2.3667	0.00693	0.0
	21	$\{15\}$	21.303	2.4114	0.00693	0.0
8	23	$\{18,20,21,24,27,29\}$	27.474	2.1449	0.00390	0.0
	24	$\{4,15,18,24,27\}$	42.796	2.3850	0.00390	0.0
	25	$\{21,26,27,29\}$	68.165	2.4500	0.00390	0.0
	26	$\{26,27,29\}$	105.39	2.4697	0.00390	0.0
	27	$\{27,29\}$	161.80	2.6175	0.00390	0.0
	28	$\{20\}$	253.16	2.6544	0.00390	0.0

APPENDIX C. EXTENDED RESULTS

This section presents a more detailed version of the results, describing the different figures obtained on the 29 tests.

C.1. **Function f_1.** Function f_1 is particularly difficult to approximate for small a_i because of the high effective dimension in the truncation sense. As shown in Figures 18 to 24, the error on estimating this function with $a_i = 0$ and $a_i = 0.01$ does not go lower than 0.01 in 20 dimensions, 0.1 in 50 dimensions and 1 in 100 dimensions. In Figure 19(b), the large spikes around 100 00 sample for AD, 200 000 samples for EA and 300 000 samples for S are due to a single randomization that has points too close to the origin. Removing those randomizations produce more conventional results as shown in Figure 21. Since we used the same 25 randomizations for each test, the same behaviour is observable in Figure 23(b), again removing those shifts makes all the sequences comparable (Figure 25). This also exlains the large irregularities of Figures 20(b) and 24(b).

As a_i increases the effective dimension in the truncation sense decreases. This is reflected in Figures 26 to 34 by a much lower estimation error. In general, all methods perform similarly on these tests.

The case $a_i = (s - i + 1)^2$ is quite interesting. Figures 35 to 37 shows that the EA sequence is significantly worse than the other sequences when the importance of the dimensions is reversed. As stated in the main document, this is probably caused by our generation method that chooses each dimension one at a time based

on the preceding dimensions, and the fact that the computational resources allowed to optimize the last dimensions is proportionally less than the preceding ones considering the combinatorial complexity of the problem.

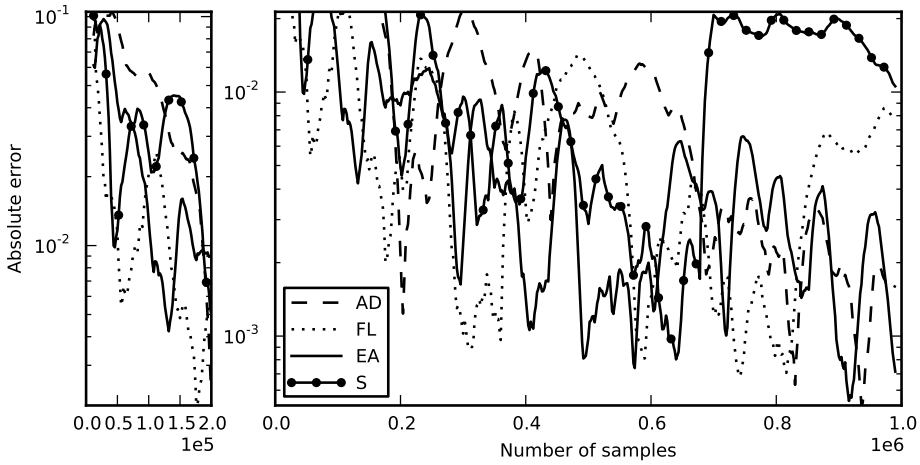
C.2. **Function f_2.** Function f_2 does provide a test that has a fairly high effective dimension in the truncation sense and a small one in the superposition sense. Figure 38 shows that neither of the tested sequences particularly suffer from this superposed effective dimension.

C.3. **Function f_3.** Function f_3 has an effective dimension in the truncation sense lower than s and in the superposition sense around 3 for $s = 20$. Figures 39 to 43 shows that all sequences perform similarly on this function. Although in Figure 41, it is notable that sequence S does not perform very well in the deterministic case with 60 dimensions, but when combined with digital shifts its performance on estimating the integral is significantly improved. Again, in Figure 42(b), the large spikes around 5 000 samples for EA and 200 000 samples for S are due to a single randomization that degenerates with an error ten times larger than all other randomizations. Removing those randomizations produce more conventional results as shown in Figure 43.

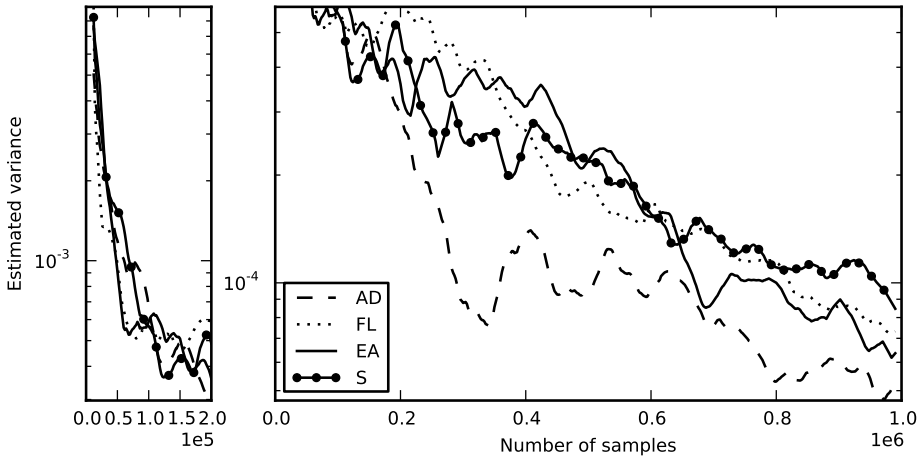
C.4. **Asian Call Option.** The last tests on the Asian call option are shown in Figures 44 to 49.

TABLE 9. Permutations of the first 20 dimensions found by the evolutionary algorithm.

s	Permutation
1	[0 1]
2	[0 2 1]
3	[0 4 2 3 1]
4	[0 6 5 4 3 2 1]
5	[0 8 2 10 4 9 5 6 1 7 3]
6	[0 10 9 5 12 3 8 4 11 7 6 2 1]
7	[0 16 11 2 12 8 7 5 15 6 3 13 9 1 10 14 4]
8	[0 8 6 1 15 7 3 16 17 11 14 18 5 9 13 4 2 10 12]
9	[0 8 1 17 13 5 16 9 6 22 18 10 21 3 12 20 4 11 19 2 14 7 15]
10	[0 21 14 8 28 2 24 10 5 27 9 26 4 25 17 1 18 7 11 19 23 15 3 20 12 22 13 6 16]
11	[0 22 15 24 7 28 5 18 20 14 16 10 25 2 11 9 19 17 29 26 4 12 23 8 1 21 30 6 13 3 27]
12	[0 23 32 3 18 22 25 15 12 8 20 34 27 19 30 6 4 33 28 36 1 10 13 9 5 35 26 29 31 21 17 16 24 7 11 14 2]
13	[0 15 39 24 11 38 1 16 32 10 20 26 12 28 27 14 5 29 6 30 21 13 19 23 18 8 25 3 36 34 31 33 22 4 35 17 2 37 9 7 40]
14	[0 34 7 17 10 25 39 40 42 16 38 18 36 27 37 15 32 11 2 24 35 41 33 3 6 23 22 28 29 30 20 8 4 19 21 31 9 1 5 12 13 14 26]
15	[0 14 9 36 38 20 41 8 25 19 28 33 46 22 30 12 37 5 35 32 44 16 10 11 3 6 23 42 17 39 18 26 4 34 43 31 21 45 27 2 7 13 1 24 15 40 29]
16	[0 7 33 48 18 1 13 47 29 9 5 14 51 3 49 32 4 16 39 12 31 28 11 17 23 40 37 30 15 21 36 52 45 35 20 41 38 43 26 24 27 10 42 50 25 19 34 2 46 22 44 8 6]
17	[0 20 44 56 17 53 12 24 13 33 19 51 40 45 1 9 36 23 18 27 41 31 47 43 57 10 38 14 2 30 46 54 6 49 34 16 11 15 29 52 25 58 8 42 50 3 35 26 39 22 48 4 55 7 37 5 28 21 32]
18	[0 22 26 5 36 60 57 59 30 12 11 13 19 29 56 58 4 35 3 25 32 43 48 16 45 33 10 23 15 50 7 14 52 8 9 6 42 46 40 47 27 34 39 2 20 41 21 54 28 44 38 49 55 53 51 18 31 37 17 24 1]
19	[0 46 63 18 59 48 58 61 50 24 3 12 15 11 14 10 32 54 7 56 49 44 6 38 53 30 37 39 36 34 5 21 52 41 66 26 64 29 9 57 60 16 17 23 55 62 19 2 43 33 8 42 47 31 65 51 40 1 20 13 28 22 4 25 45 27 35]
20	[0 63 32 27 31 67 49 43 51 56 22 55 66 29 11 65 60 8 39 37 40 18 2 30 24 36 35 48 17 16 58 62 12 26 47 3 54 33 68 34 28 1 46 69 53 61 42 59 13 25 14 5 64 4 23 15 45 20 10 9 44 41 21 38 57 52 7 19 70 6 50]

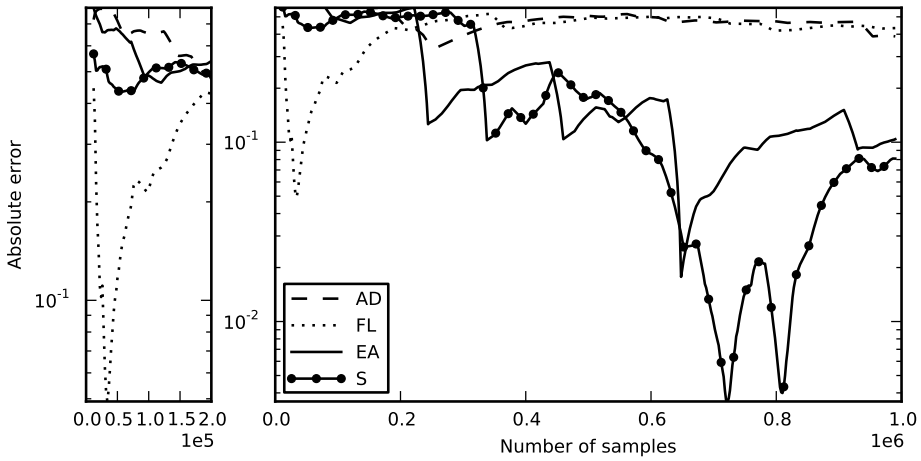


(a)

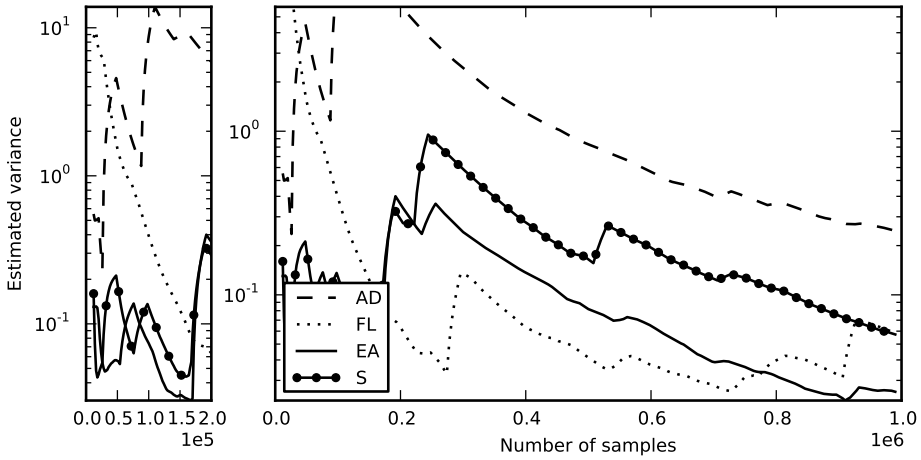


(b)

FIGURE 18. Test function f_1, with $a_i = 0$ and $s = 20$.

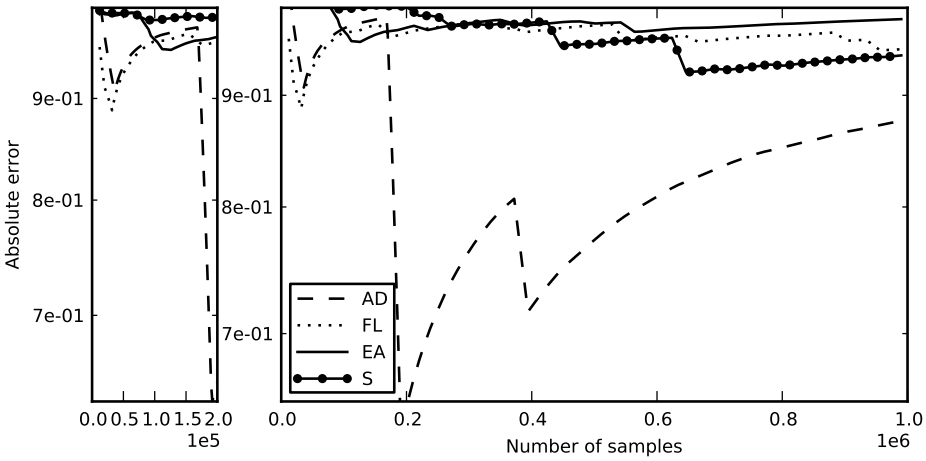


(a)

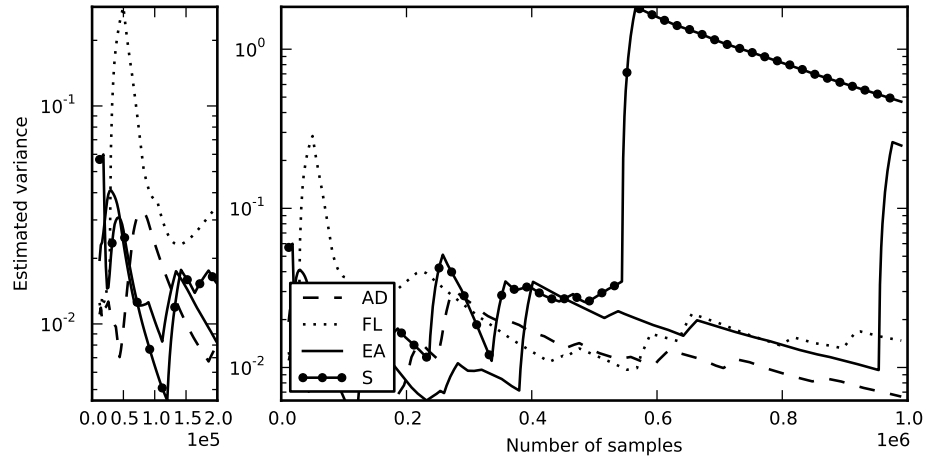


(b)

FIGURE 19. Test function f_1, with $a_i = 0$ and $s = 50$.



(a)



(b)

FIGURE 20. Test function f_1, with $a_i = 0$ and $s = 100$

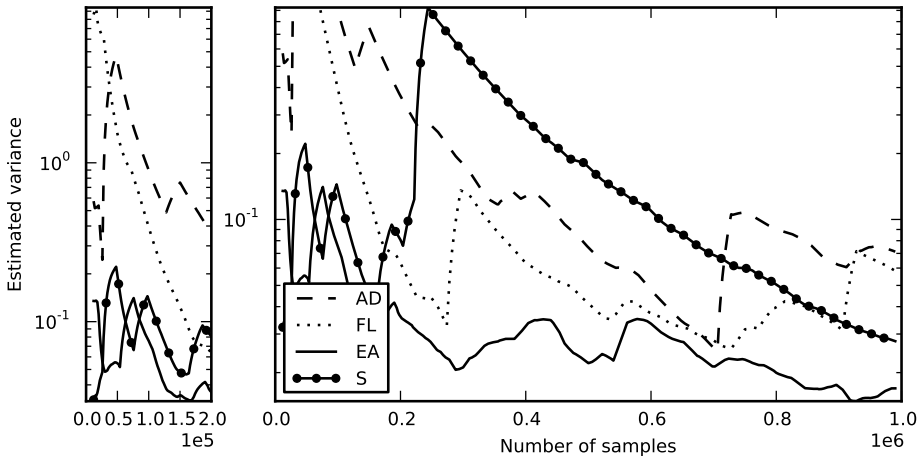
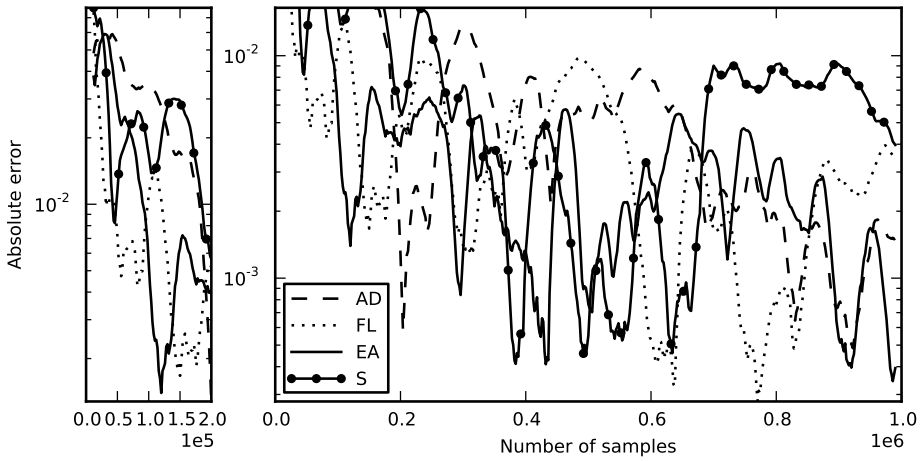
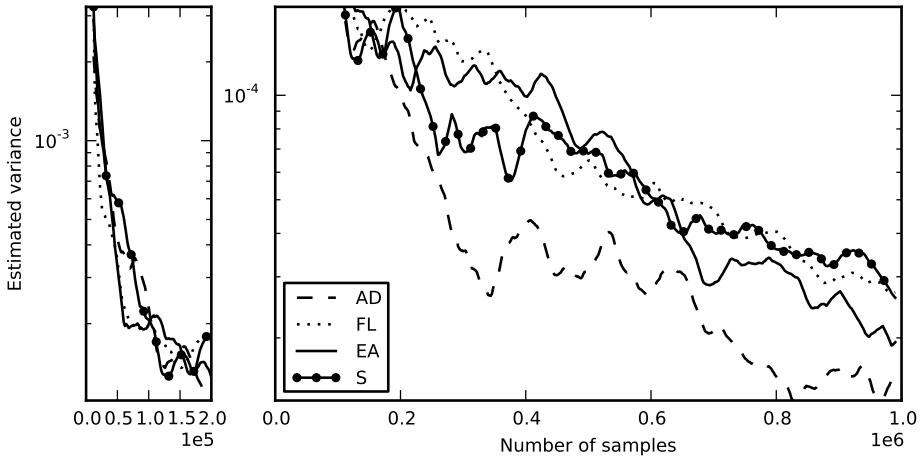


FIGURE 21. Test function f_1, with $a_i = 0$, $s = 50$ and degenerated sequences removed.

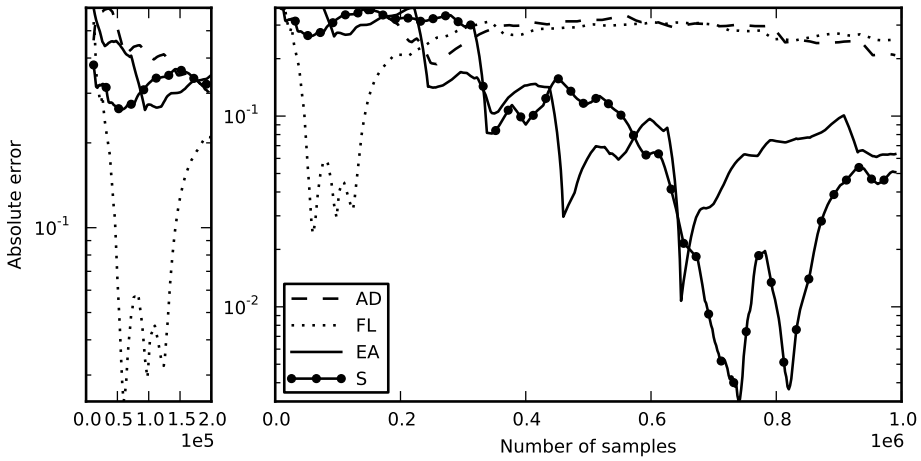


(a)

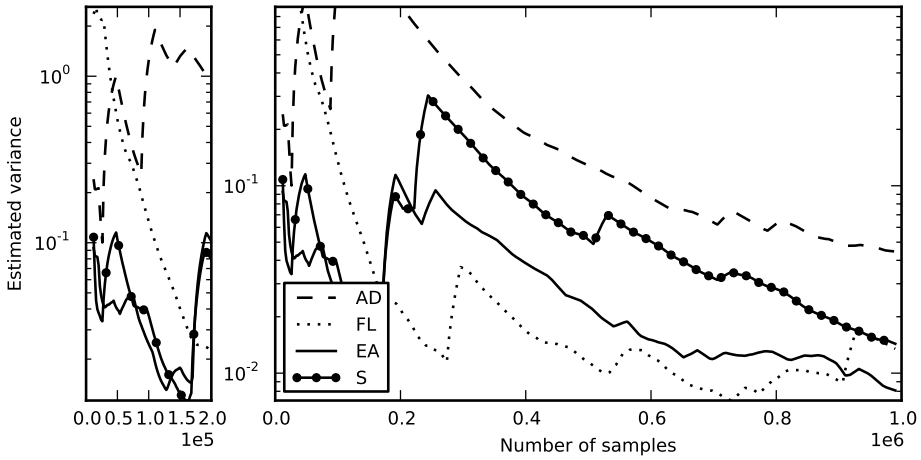


(b)

FIGURE 22. Test function f_1, with $a_i = 0.01$ and $s = 20$

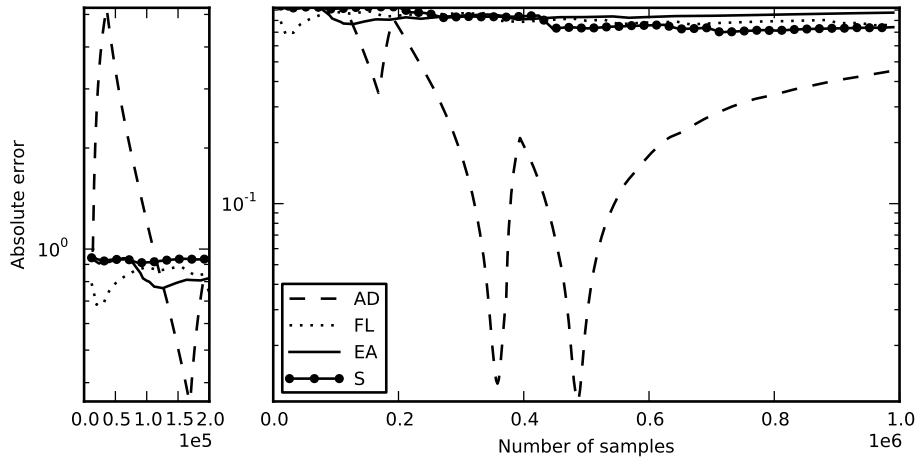


(a)

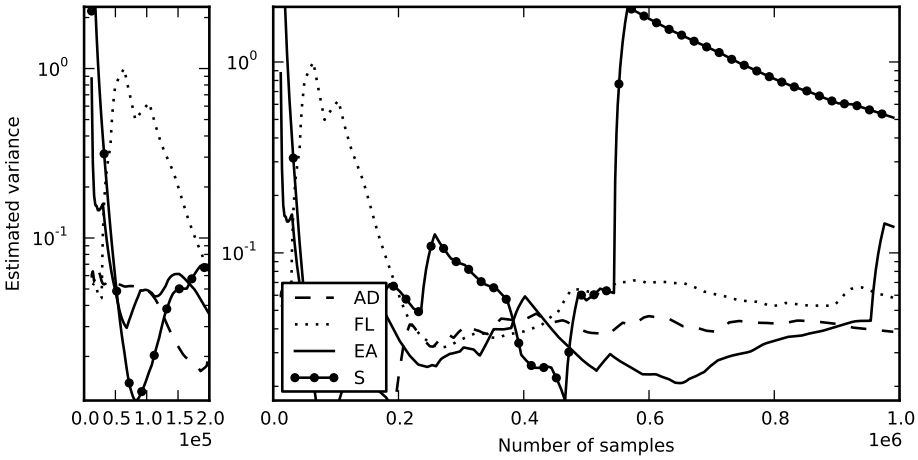


(b)

FIGURE 23. Test function f_1, with $a_i = 0.01$ and $s = 50$



(a)



(b)

FIGURE 24. Test function f_1, with $a_i = 0.01$ and $s = 100$

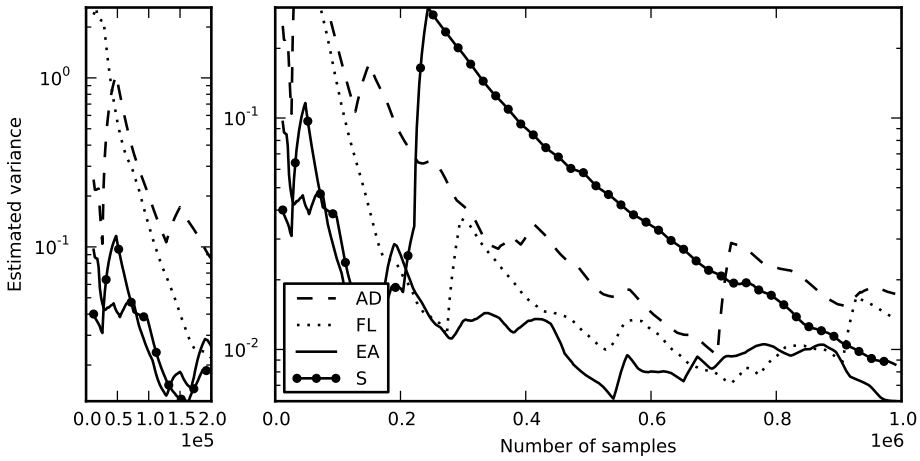
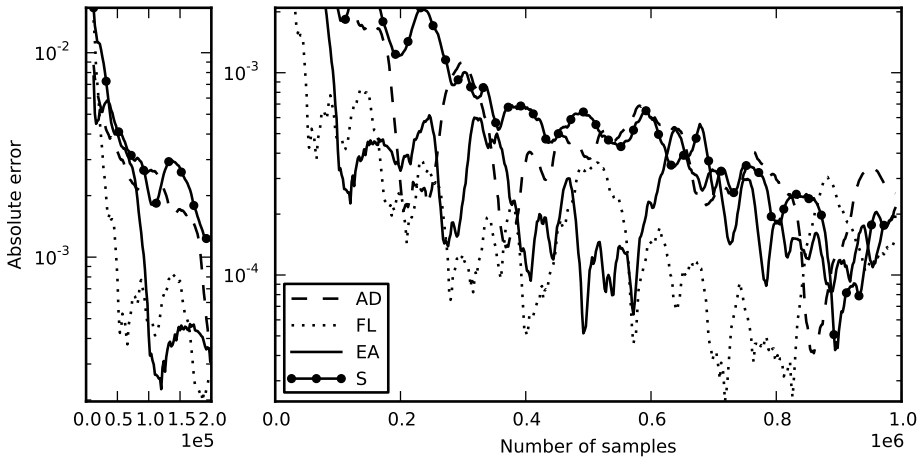
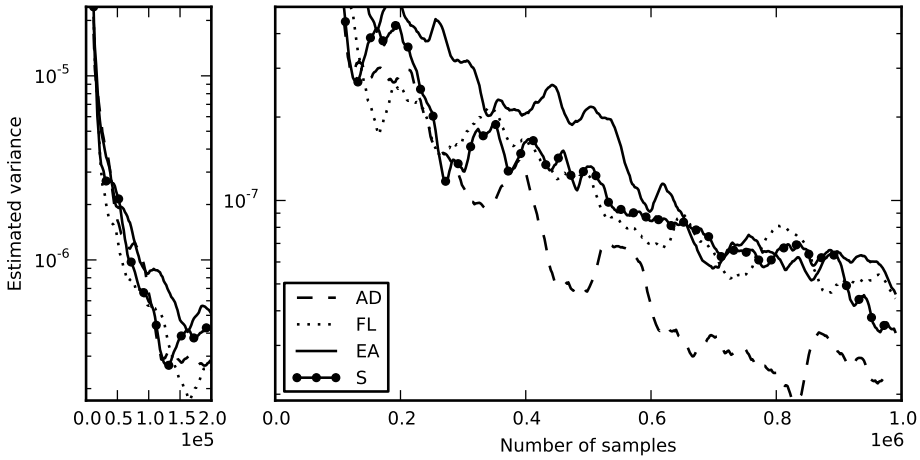


FIGURE 25. Test function f_1, with $a_i = 0.01$, $s = 50$ and degenerated sequences removed.

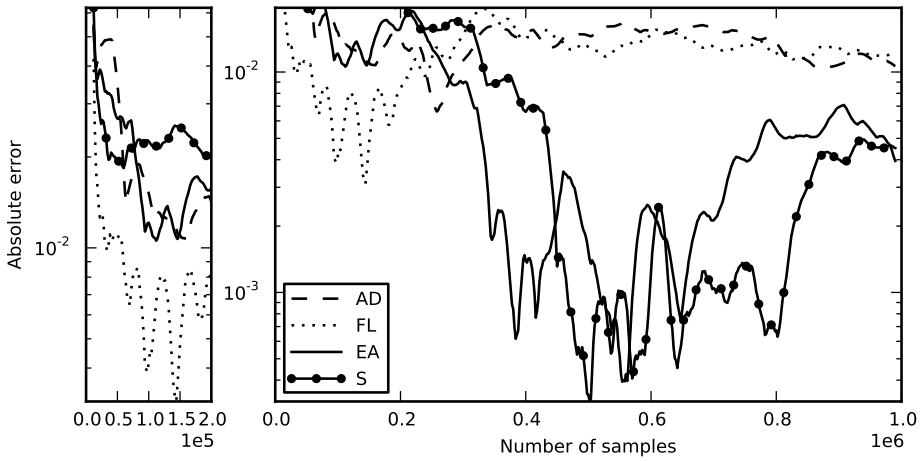


(a)

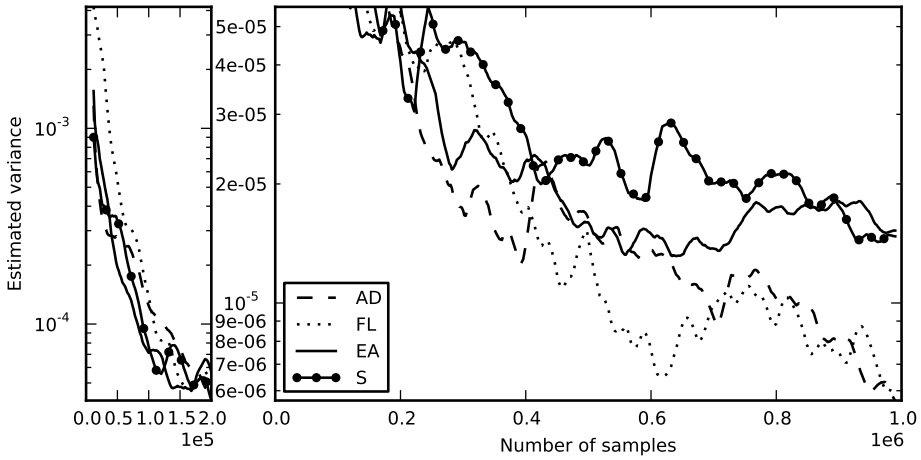


(b)

FIGURE 26. Test function f_1, with $a_i = 1.0$ and $s = 20$

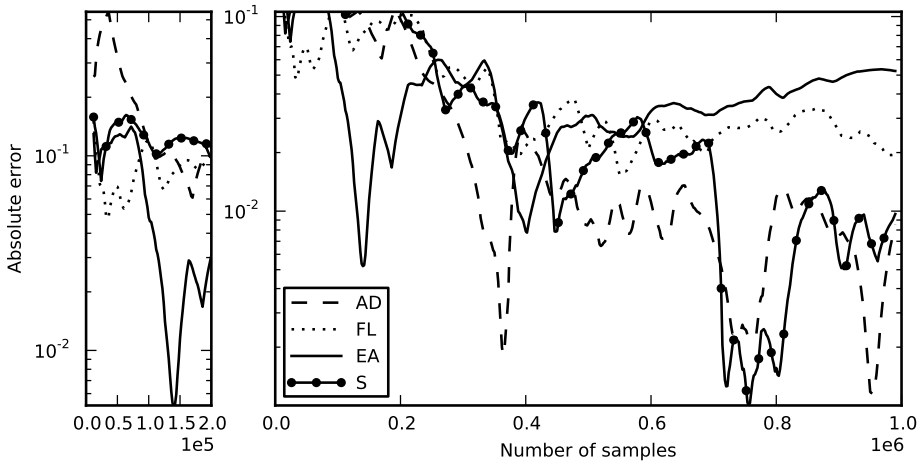


(a)

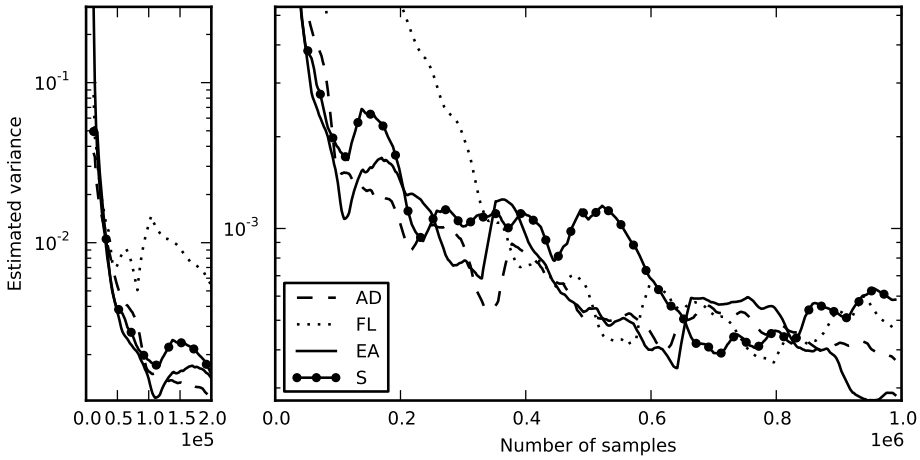


(b)

FIGURE 27. Test function f_1, with $a_i = 1.0$ and $s = 50$

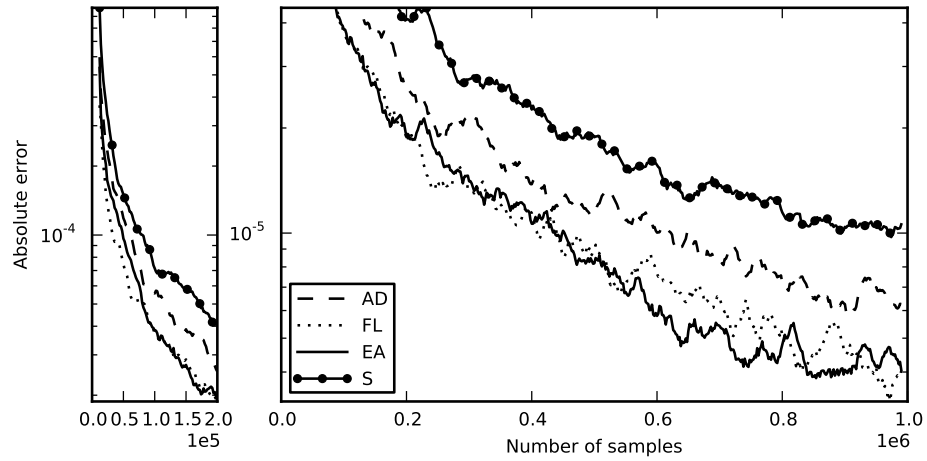


(a)

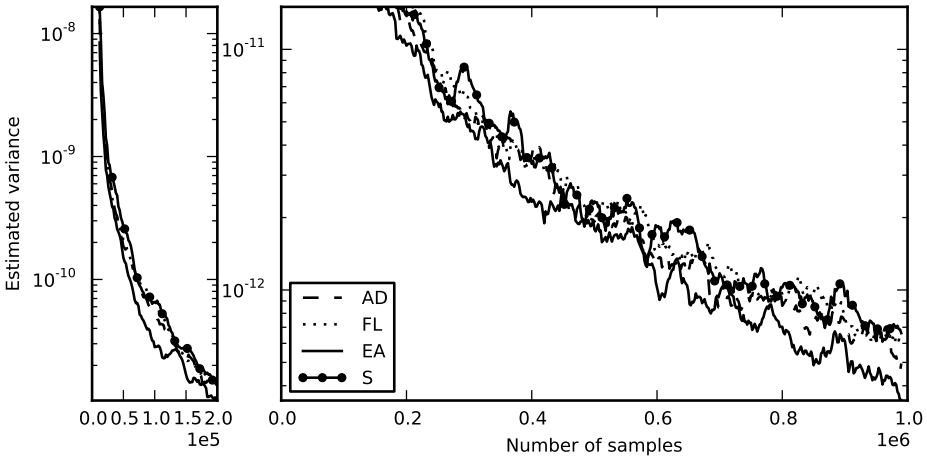


(b)

FIGURE 28. Test function f_1, with $a_i = 1.0$ and $s = 100$

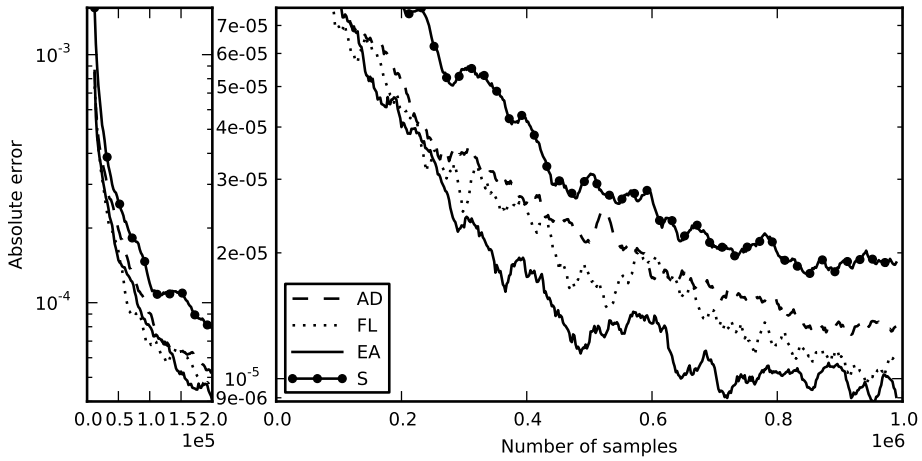


(a)

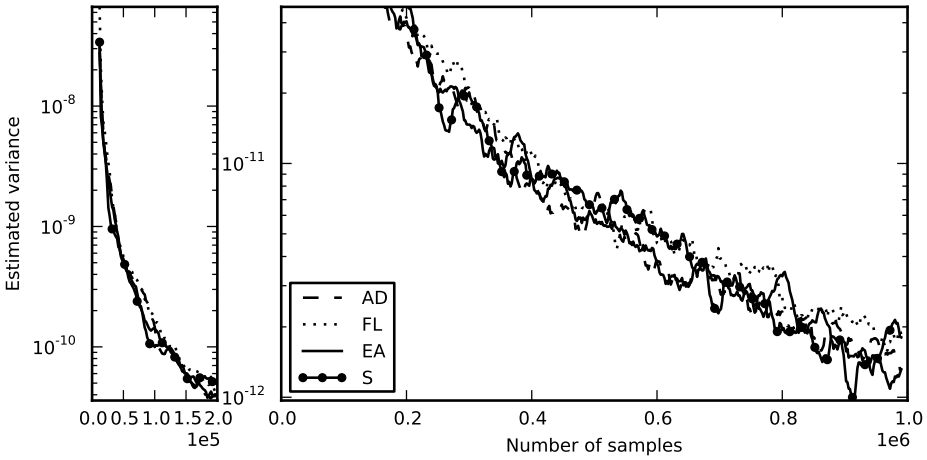


(b)

FIGURE 29. Test function f_1, with $a_i = i$ and $s = 20$

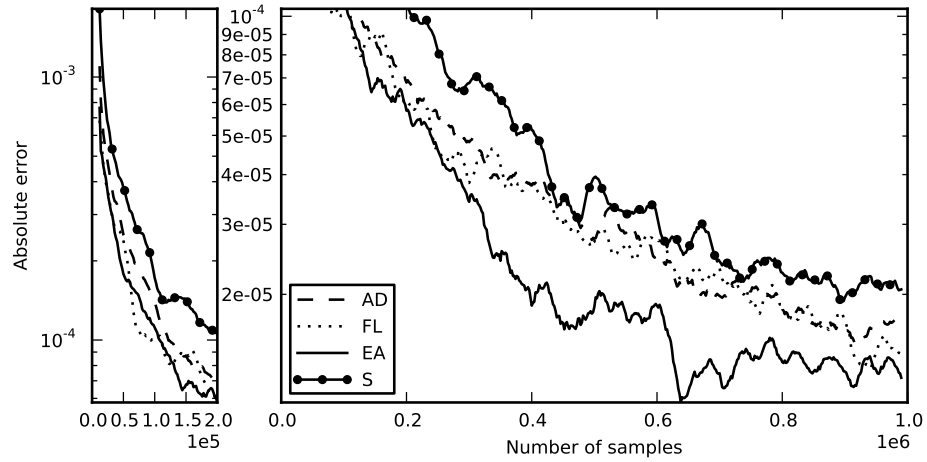


(a)

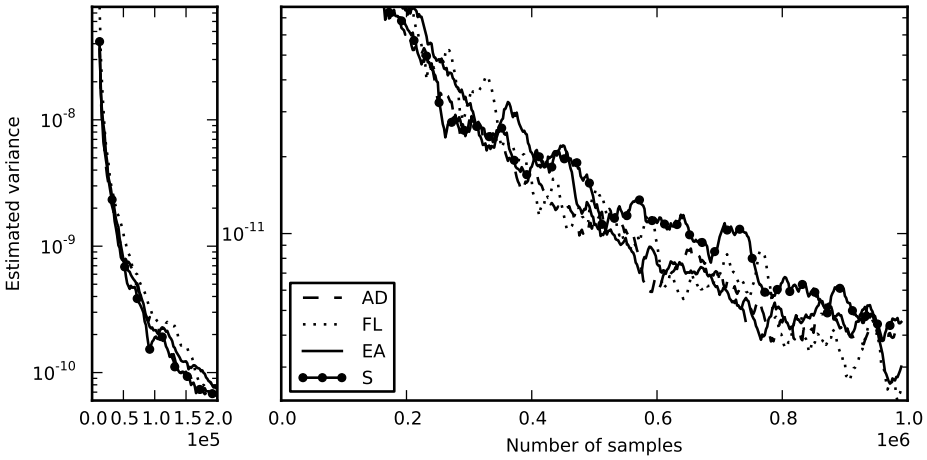


(b)

FIGURE 30. Test function f_1, with $a_i = i$ and $s = 50$

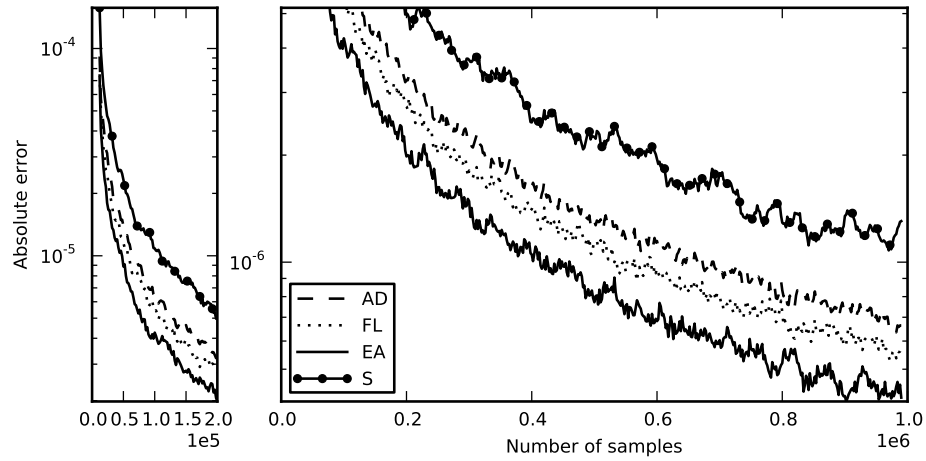


(a)

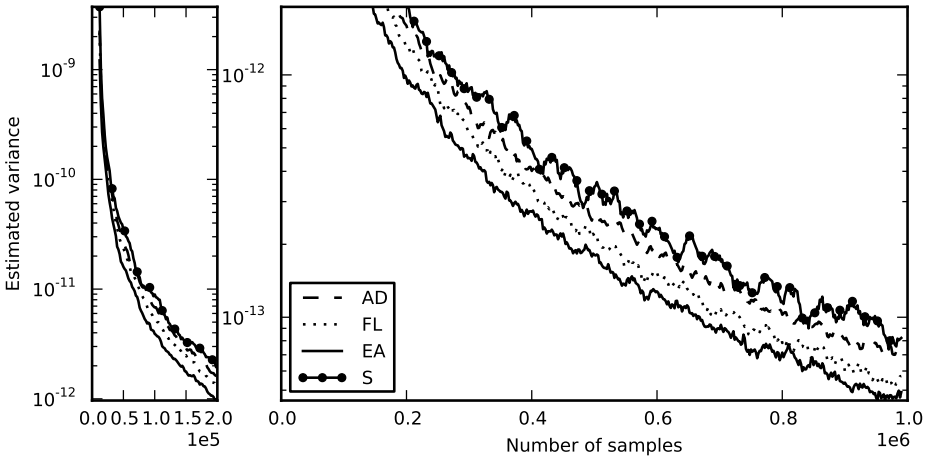


(b)

FIGURE 31. Test function f_1, with $a_i = i$ and $s = 100$

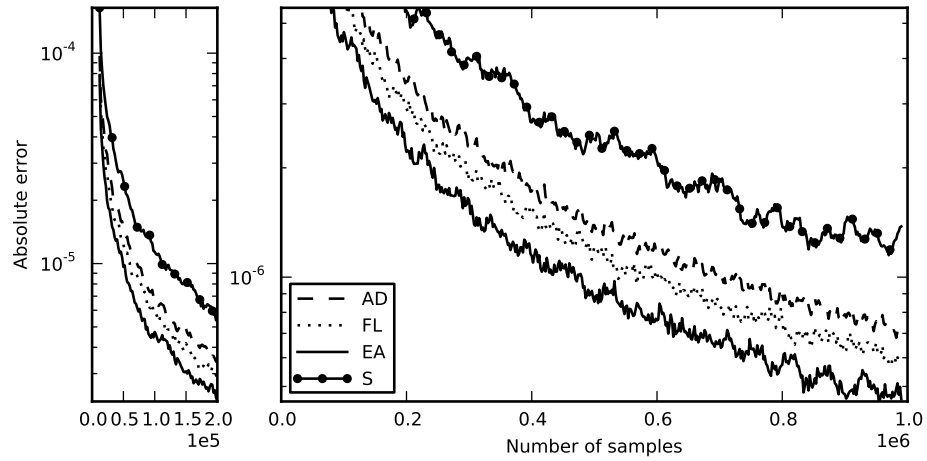


(a)

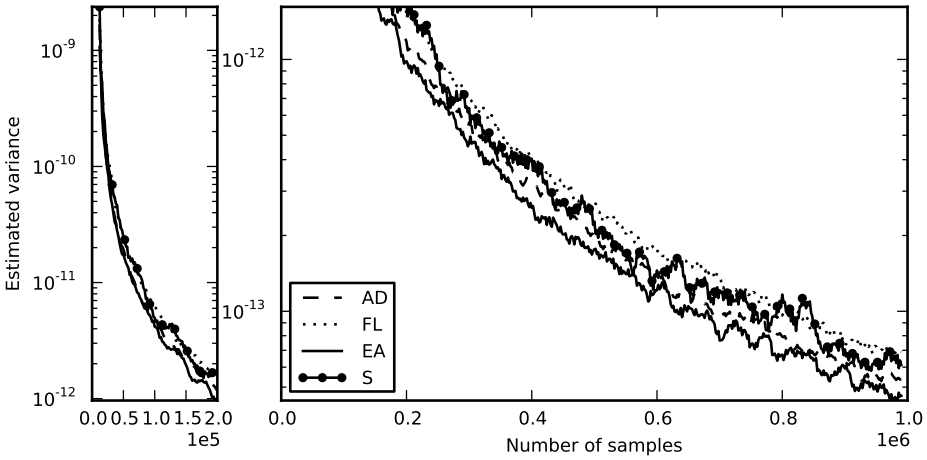


(b)

FIGURE 32. Test function f_1, with $a_i = i^2$ and $s = 20$

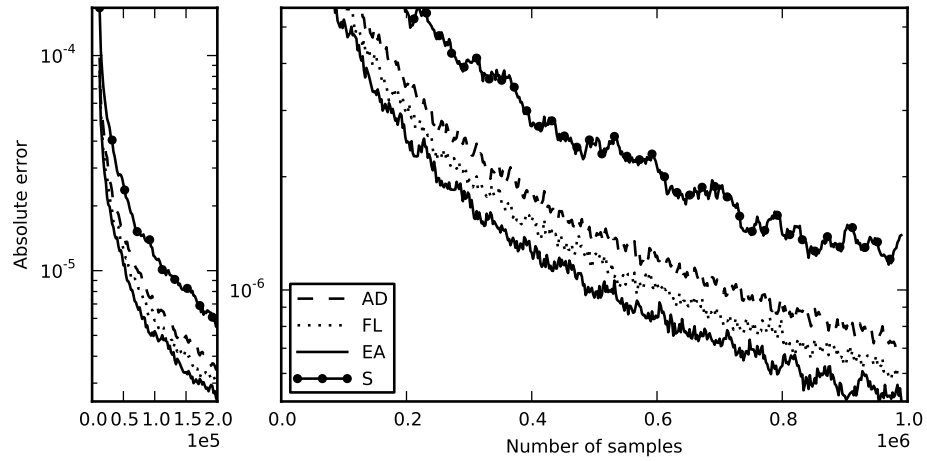


(a)

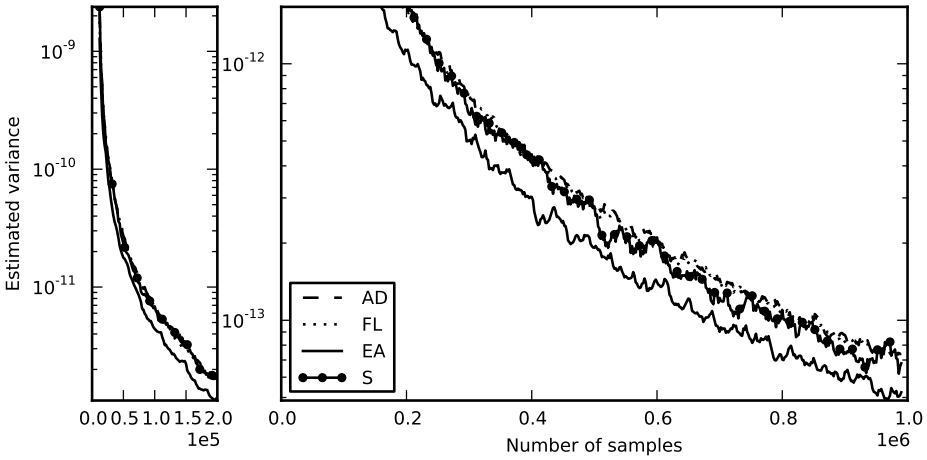


(b)

FIGURE 33. Test function f_1, with $a_i = i^2$ and $s = 50$

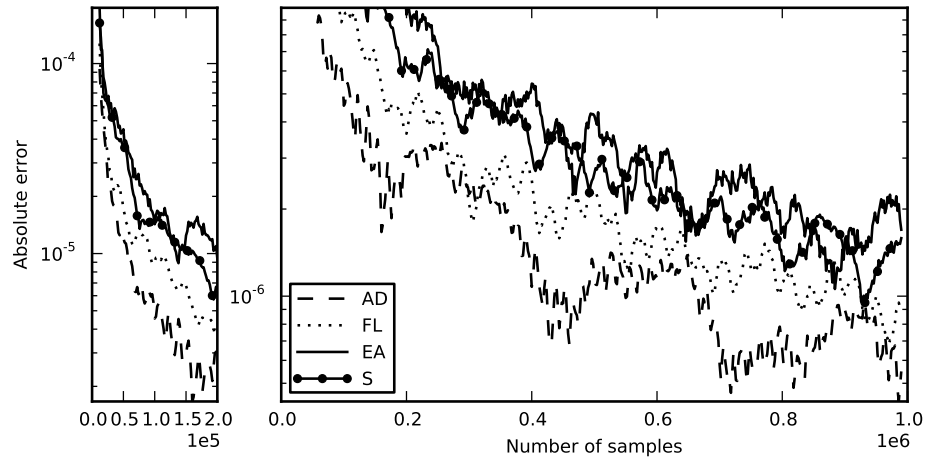


(a)

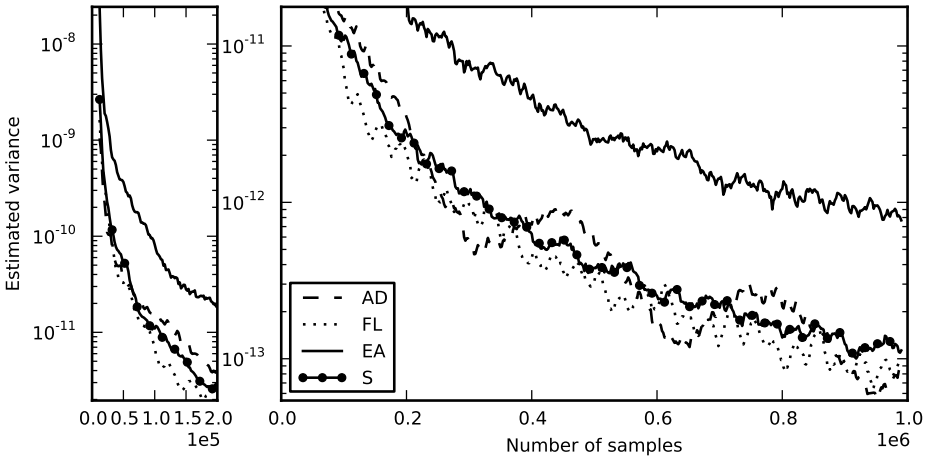


(b)

FIGURE 34. Test function f_1, with $a_i = i^2$ and $s = 100$

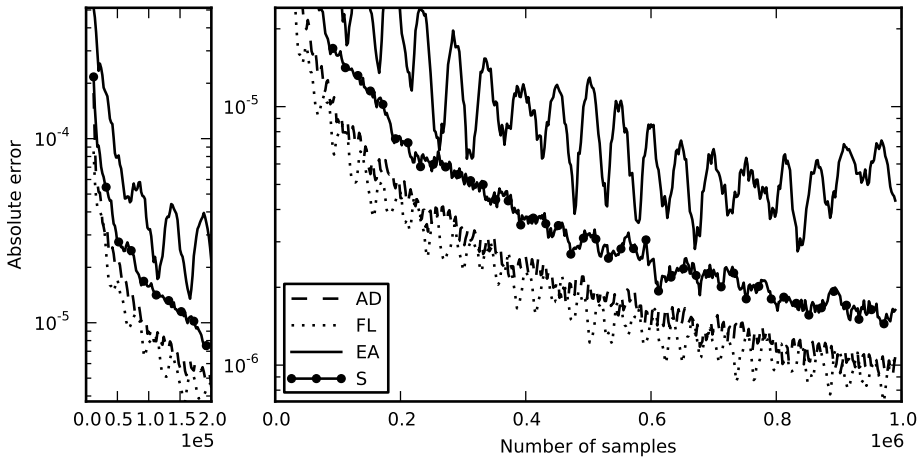


(a)

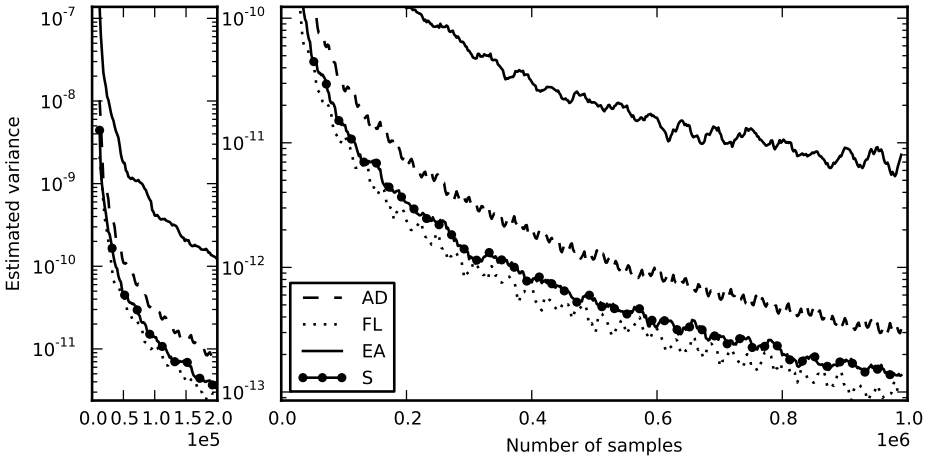


(b)

FIGURE 35. Test function f_1, with $a_i = (s - i + 1)^2$ and $s = 20$

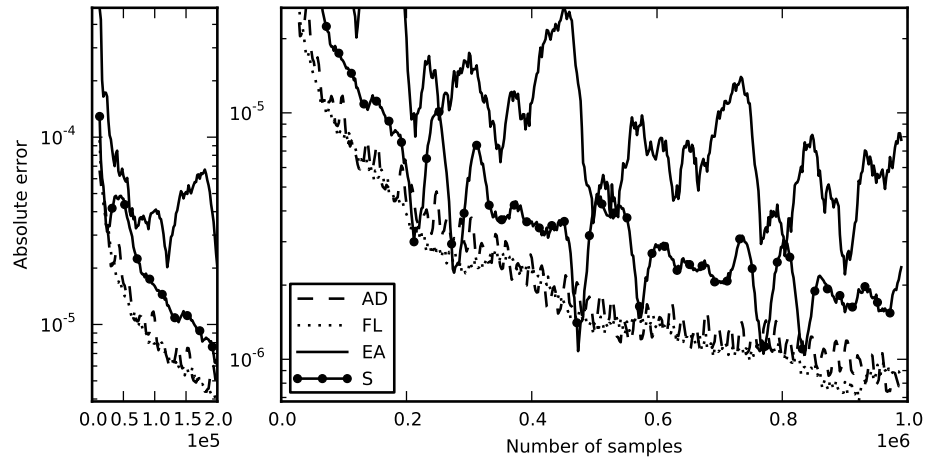


(a)

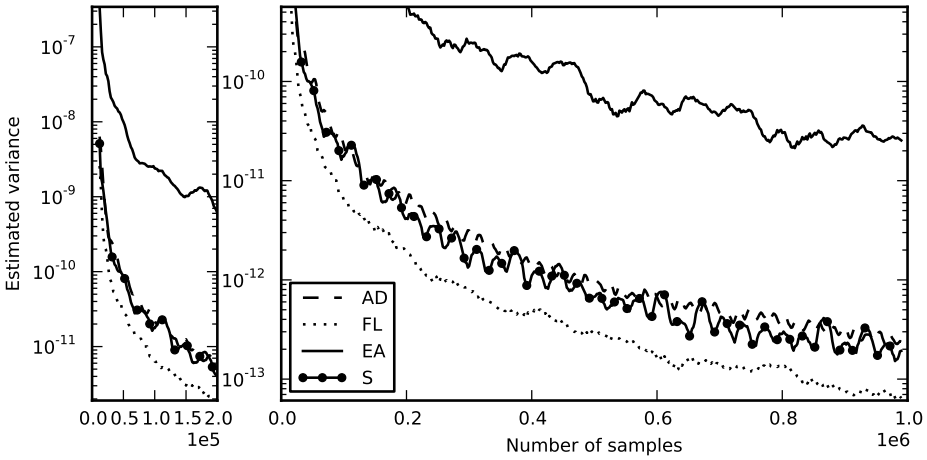


(b)

FIGURE 36. Test function f_1, with $a_i = (s - i + 1)^2$ and $s = 50$

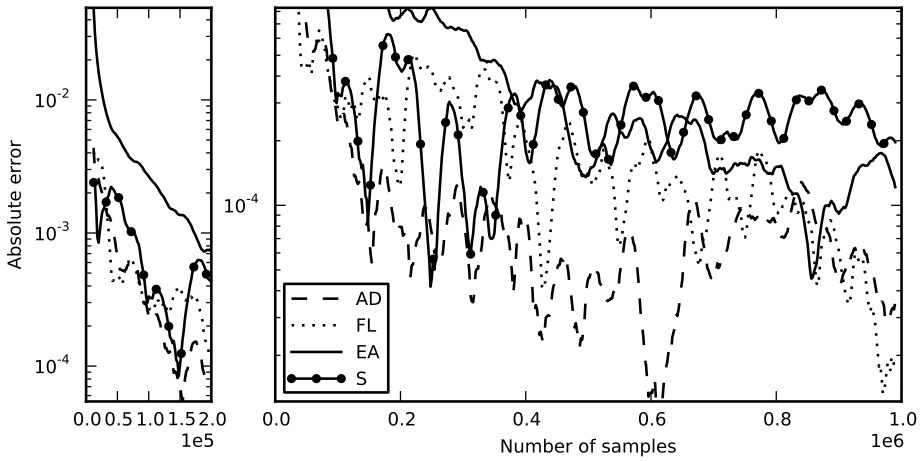


(a)

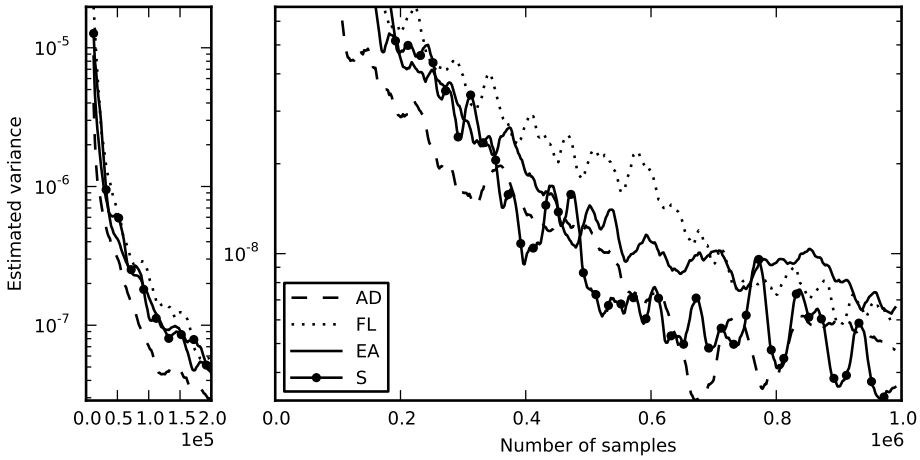


(b)

FIGURE 37. Test function f_1, with $a_i = (s - i + 1)^2$ and $s = 100$

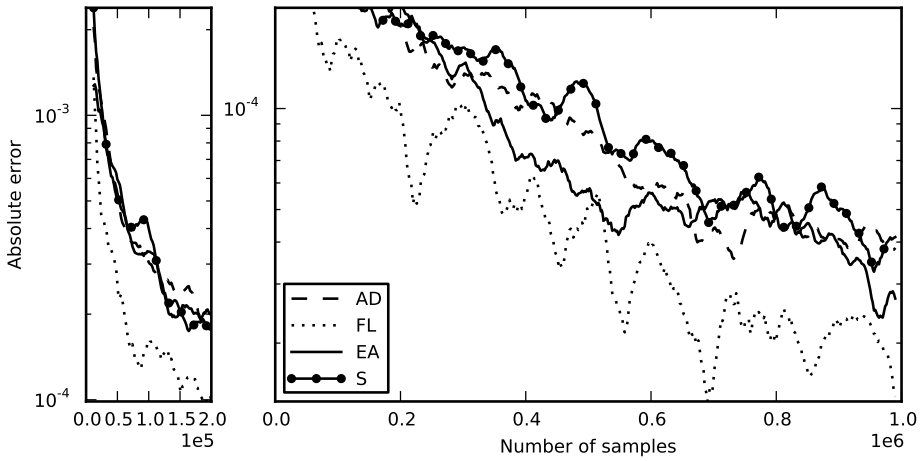


(a)

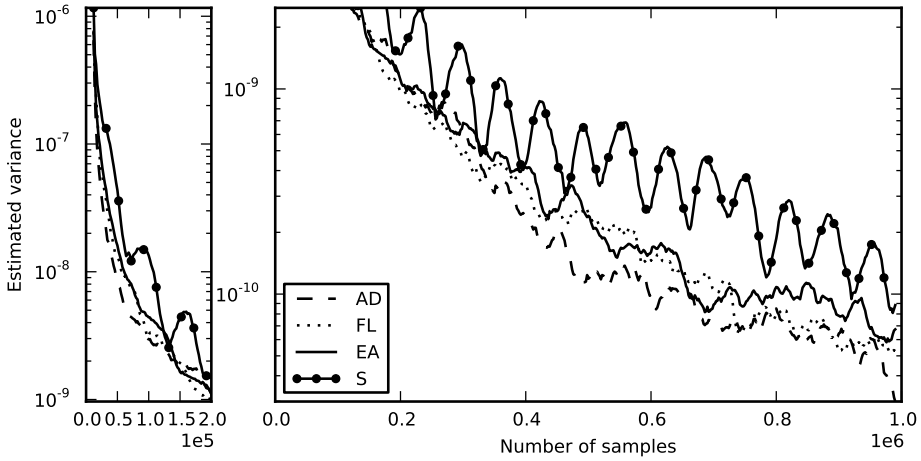


(b)

FIGURE 38. Test function f_2, with $c = 0.25$ and $s = 96$

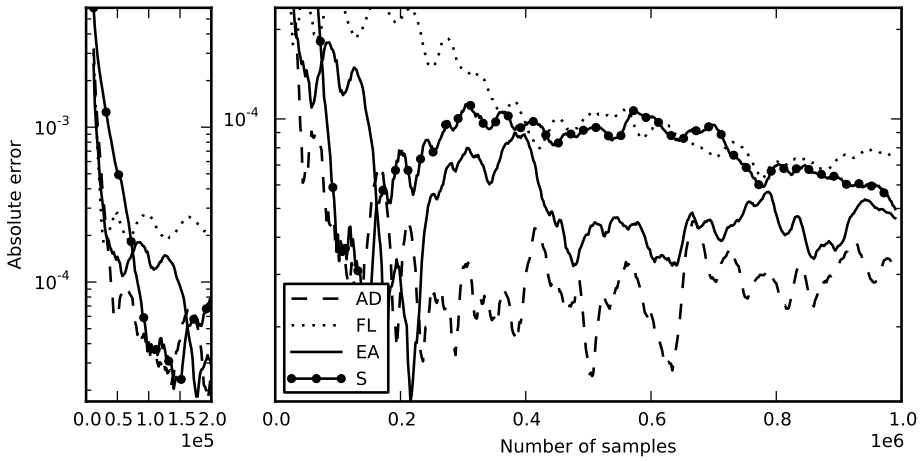


(a)

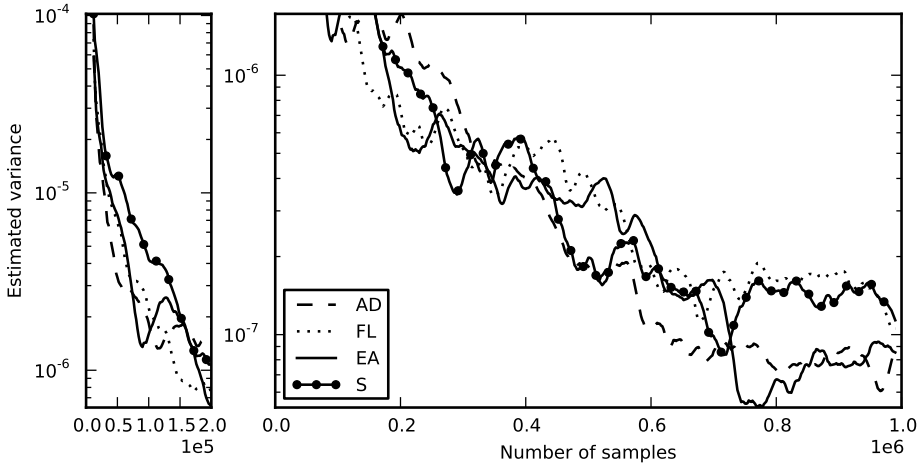


(b)

FIGURE 39. Test function f_3, with $s = 9$

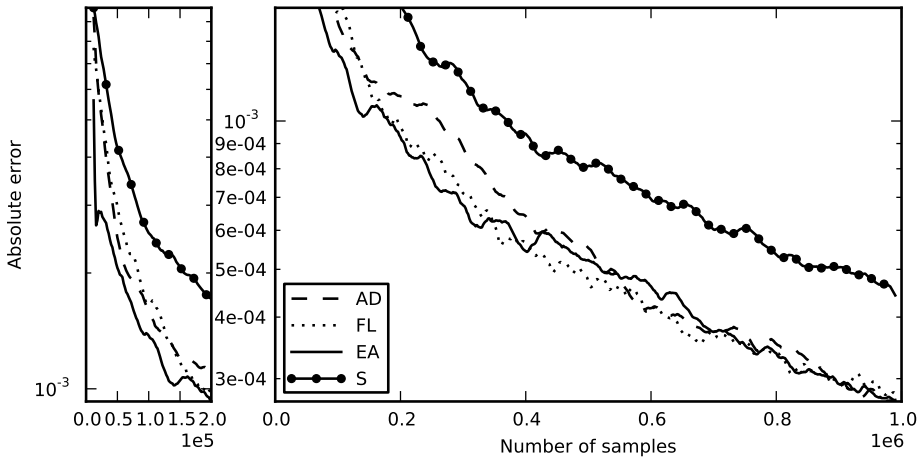


(a)

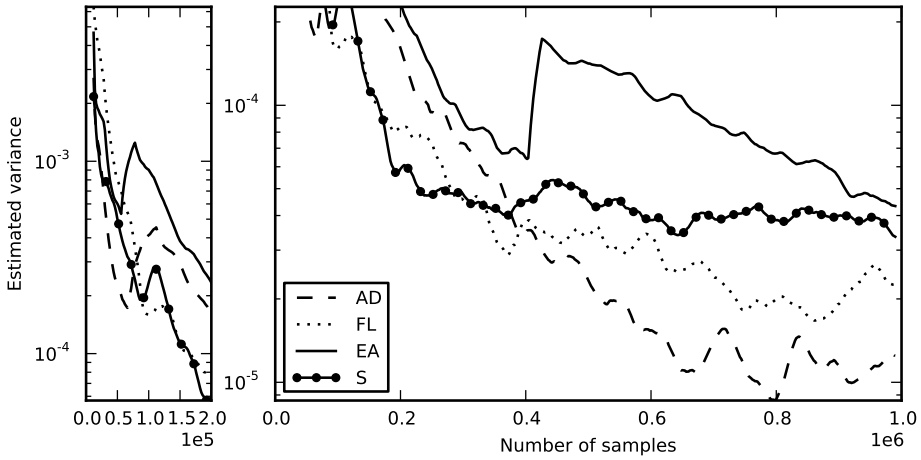


(b)

FIGURE 40. Test function f_3, with $s = 25$

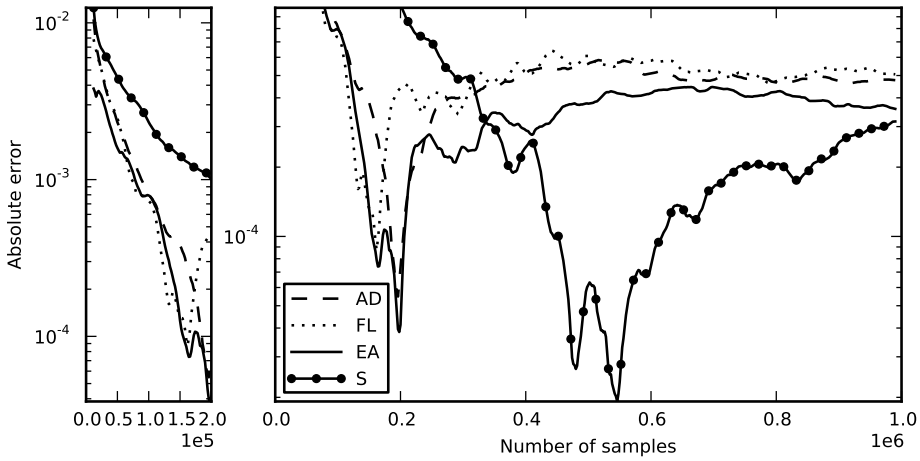


(a)

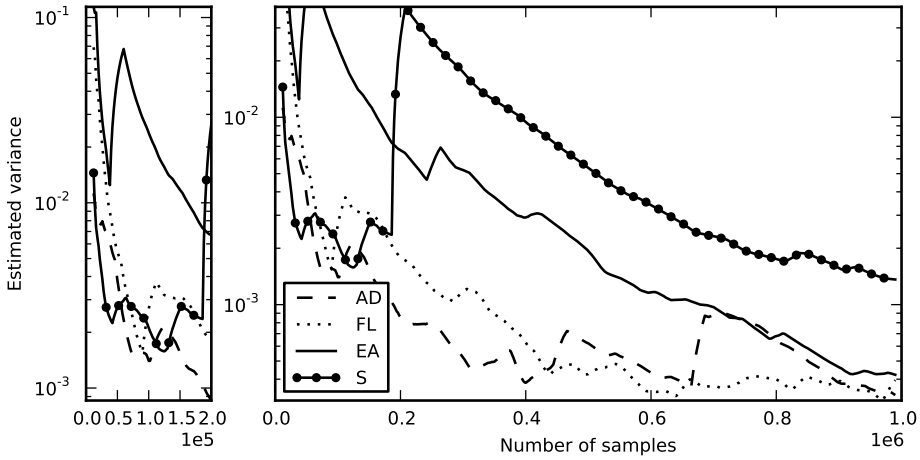


(b)

FIGURE 41. Test function f_3, with $s = 60$



(a)



(b)

FIGURE 42. Test function f_3, with $s = 100$

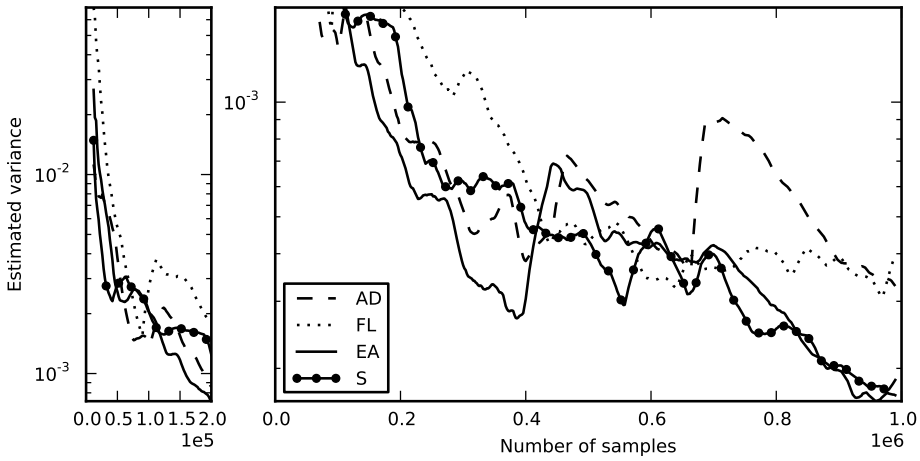
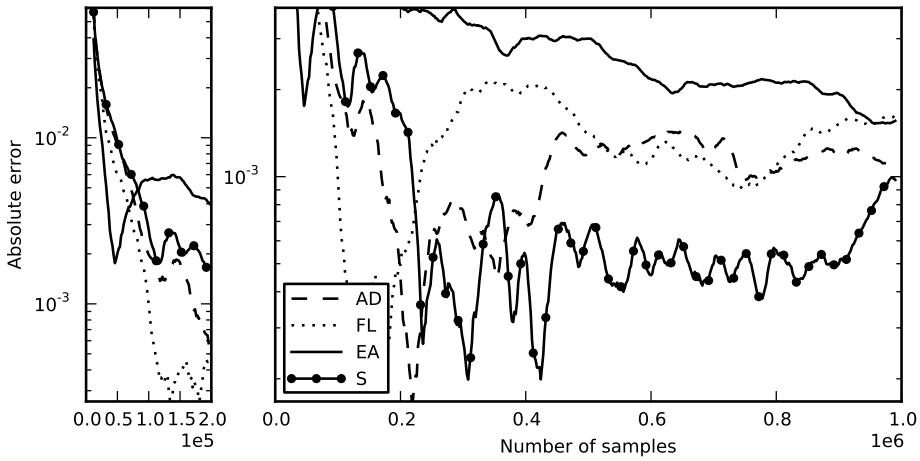
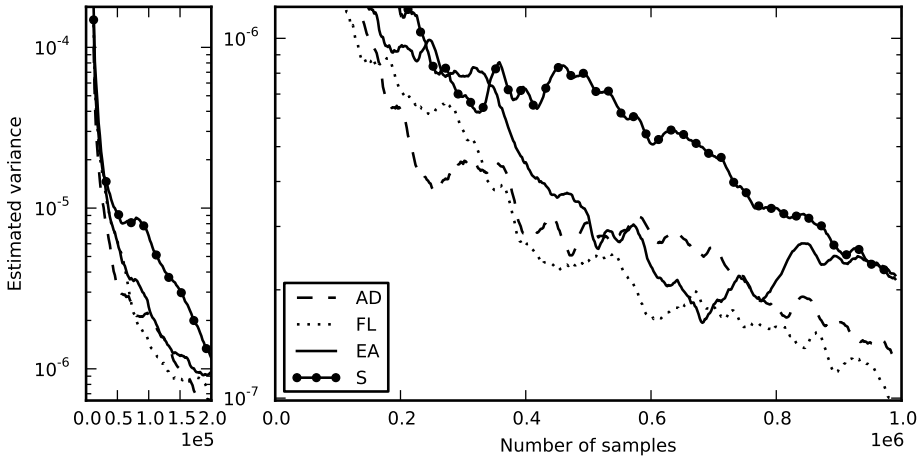


FIGURE 43. Test function f_3, with $s = 100$ and degenerated sequences removed.

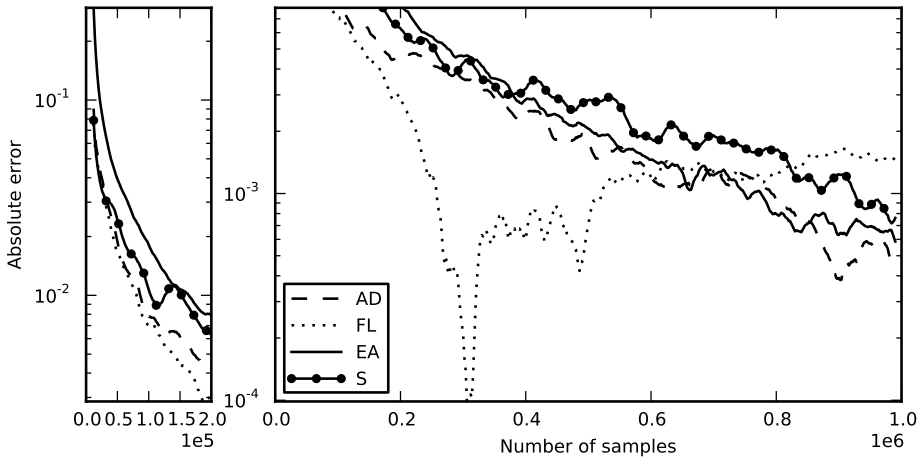


(a)

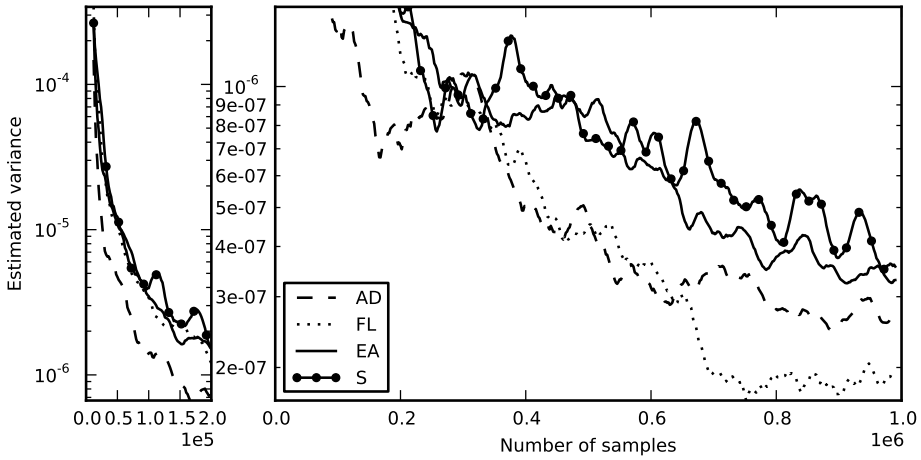


(b)

FIGURE 44. Asian call option, with $K = 45$ and $s = 40$

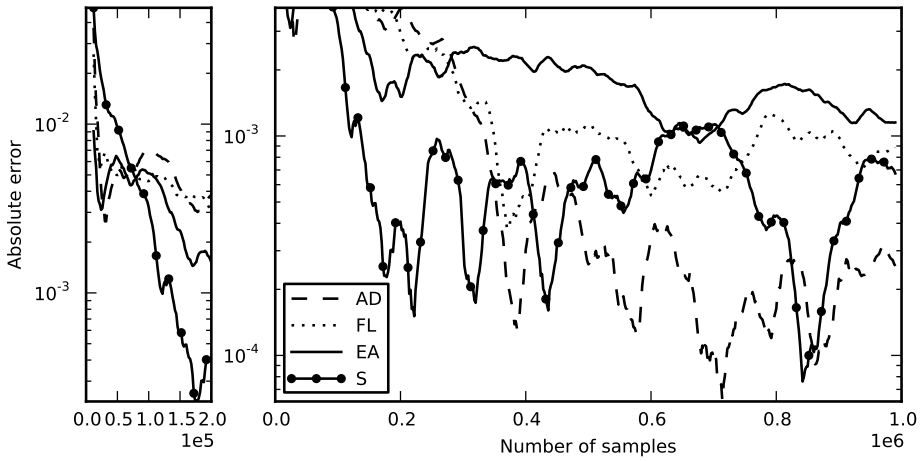


(a)

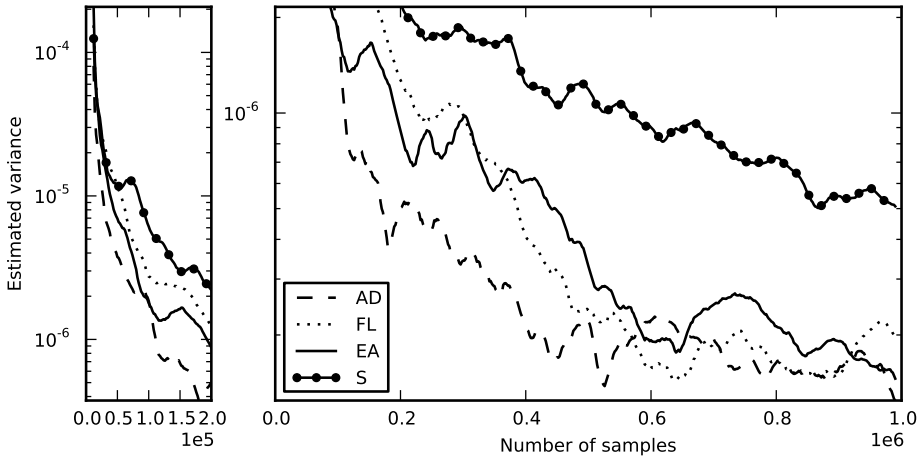


(b)

FIGURE 45. Asian call option, with $K = 45$ and $s = 75$

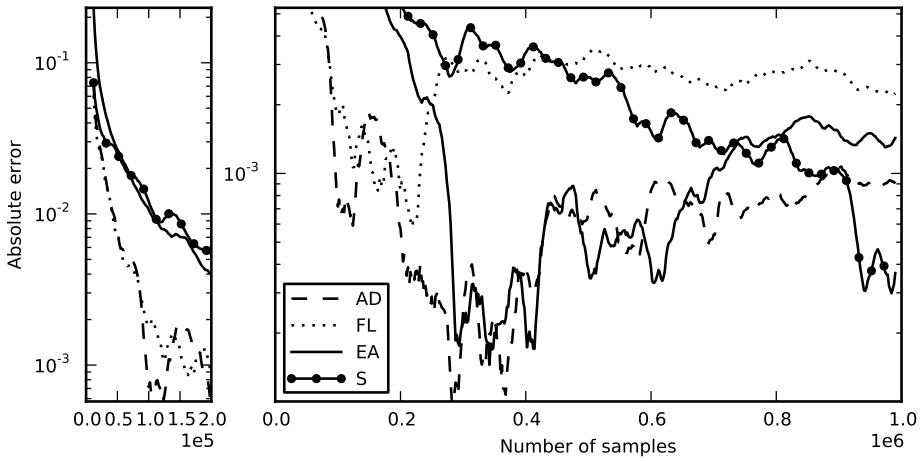


(a)

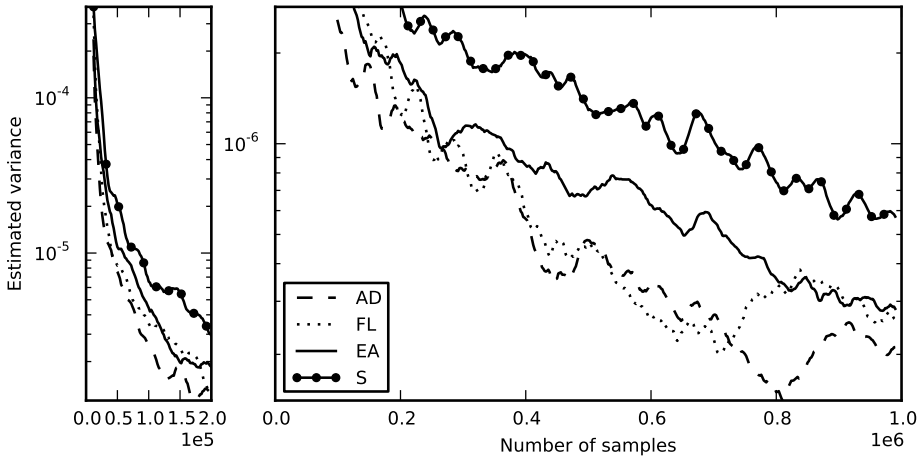


(b)

FIGURE 46. Asian call option, with $K = 50$ and $s = 40$

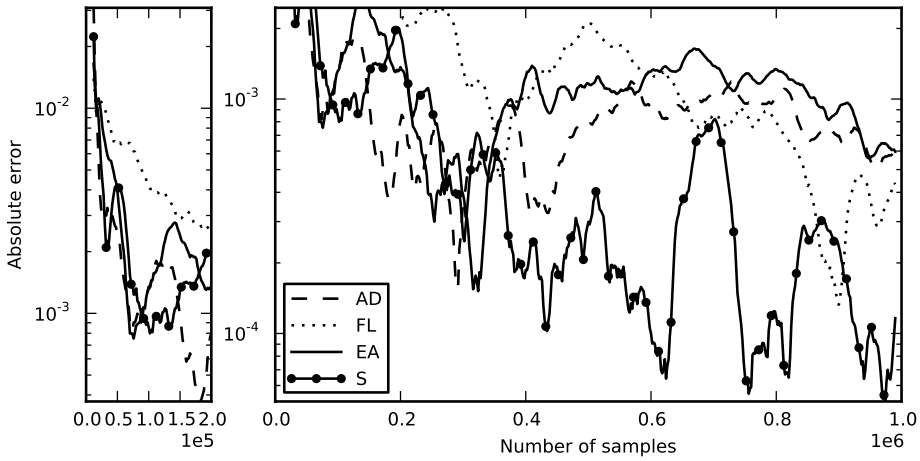


(a)

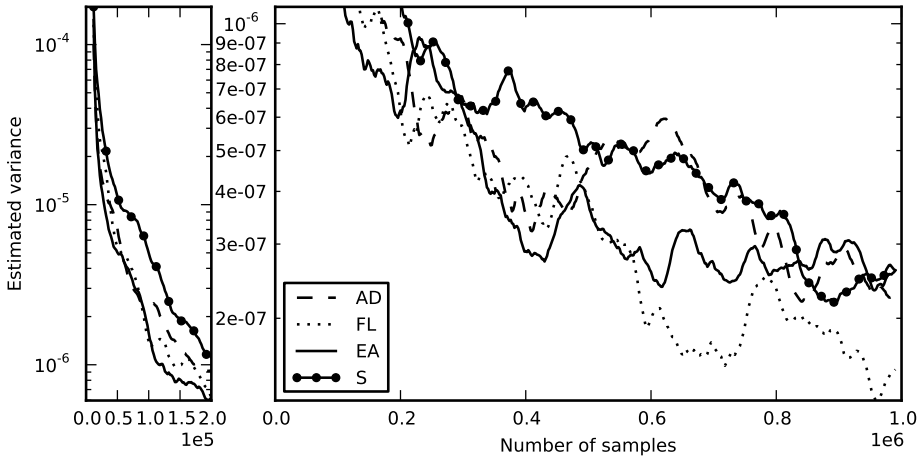


(b)

FIGURE 47. Asian call option, with $K = 50$ and $s = 75$

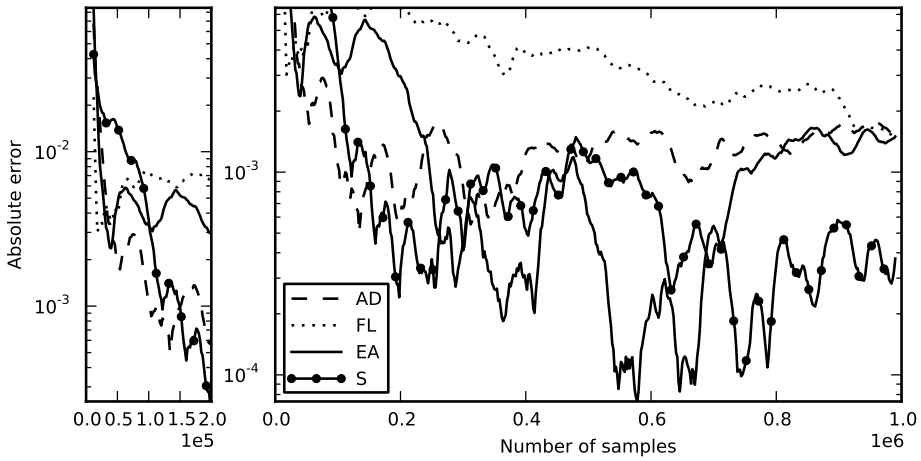


(a)

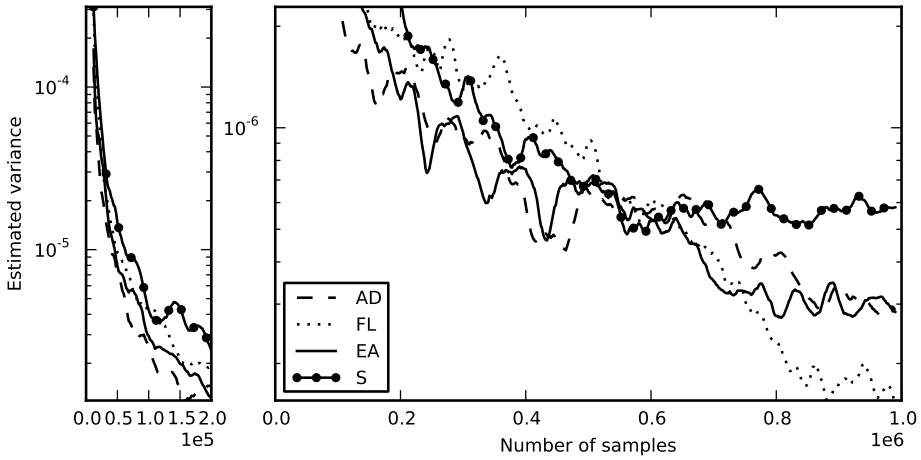


(b)

FIGURE 48. Asian call option, with $K = 55$ and $s = 40$



(a)



(b)

FIGURE 49. Asian call option, with $K = 55$ and $s = 75$